高等教育计算机学科“应用型”教材

Linux 操作系统（第 3 版）

邵国金　主　编

褚龙现　张　娜　副主编

郭玉东　主　审

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

VISIT…

LANZAROTE

Caliente.COM

内 容 简 介

本书以红帽（Fedora 24、CentOS 7 和 RHEL 7.3）及 Ubuntu 16.04 系统为蓝本，分 4 篇介绍了 Linux 系统的基本操作、管理、编程和网络服务及应用。基础篇介绍了 UNIX/Linux 系统的基本知识和基本操作；系统管理篇介绍了 UNIX/Linux 系统的常用管理，内容包括用户、组和密码管理，文件系统管理及使用，进程、任务与作业管理，系统安装、扩充、启动与管理，设备管理，网络管理与网络应用及 SELinux、防火墙与系统安全等；编程与开发篇介绍了 bshell 编程、C/C++编程和 Java、Python 等其他编程环境；网络服务与应用篇介绍了 Linux 系统的常用网络应用与网络服务，包括网络基础服务 SSH 和 VNC、时间服务 NTP、Web 服务器 Apache、DHCP 服务器、FTP 与 TFTP 服务器、网络资源共享服务 Samba 和 NFS、域名服务器 DNS 等。

本书与以前版本的最大不同是引进了 Linux 系统的最新技术和应用，强化了安全技术 SELinux 与防火墙。

本书从培养“应用型”人才出发，兼顾基本知识和基本理论，内容翔实，结构清晰，具有较强的实用性和指导性；基于不断发展，安全稳定，为虚拟化、大数据和云计算提供最好支持的红帽和 Ubuntu 系统，具有广泛的代表性，且内容兼顾 UNIX。

本书可作为高等院校 UNIX/Linux 操作系统的教材，也可作为网络操作系统的实例教材，更可作为 UNIX/Linux 操作系统管理者和爱好者的参考书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

Linux 操作系统/邵国金主编. —3 版. —北京：电子工业出版社，2018.1

ISBN 978-7-121-33046-9

Ⅰ. ①L… Ⅱ. ①邵… Ⅲ. ①Linux 操作系统－高等学校－教材 Ⅳ. ①TP316.89

中国版本图书馆 CIP 数据核字（2017）第 281805 号

策划编辑：张贵芹
责任编辑：康 霞
印 刷：
装 订：
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1092 1/16 印张：31.25 字数：800 千字
版 次：2008 年 1 月第 1 版
2018 年 1 月第 3 版
印 次：2018 年 1 月第 1 次印刷
定 价：59.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）88254511；82470125@qq.com。

前　　言

关于本书

Linux 是计算机爱好者的操作系统，也是专业人士的操作系统。这不仅因为她有一个自由的、开放源代码的王国，更因为她在 IT 新技术、新应用中有不俗的表现。通过学习 Linux，可使计算机爱好者掌握核心技术，成为计算机或操作系统的高手或专业人士。

UNIX 是 Linux 系统的前身，Linux 是对 UNIX 系统的发展，从某种意义上讲 Linux 就是某个版本的 UNIX，因此 Linux 像 UNIX 系统一样具有可靠、高效和稳定性等特点。从 Linux 到处都可看到 UNIX 系统的身影，因此可以说，学习了 Linux 就等于学习了 UNIX。

作为一个操作系统，Linux 早已涉足政府办公、军事战略和商业运行的方方面面，在电子政务、电子商务、网站建设、嵌入式、移动开发、虚拟化、云计算、大数据等众多领域大显身手，但我们必须承认，作为操作系统的 Linux，比 Windows 等具有更高的专业性，因此就管理和使用来讲，对用户有更高的要求。这也是编写本书的出发点之一。

本书是基于红帽（Fedora 24、CentOS 7 和 RHEL 7.3）及 Ubuntu（16.04）系统的，它们都具有开放、创新、前瞻性等特点，可运行的体系结构包括 X86、X86_64 和 PowerPC 等多种平台。由于 Linux 发展迅速，图形界面在不同版本间差别很大，所以本书主要侧重于字符、命令界面和基本知识、基本技能的介绍，并兼顾 UNIX 和 Linux。

本书结构

全书分为 4 篇，共 19 章，大致内容如下。

基础篇：包含系统简介、入门和 shell 与 shell 基本操作三章。通过对本篇的学习，用户可以掌握 Linux 基本知识和基本操作，并能处理一般性问题。

系统管理篇：包括用户、组和密码管理，文件系统管理及使用，进程、任务与作业管理，系统安装、扩充、启动与管理，设备管理，网络管理与网络基本应用和 Linux 系统安全等章节。本篇是传统 UNIX 和现代 Linux 系统的管理核心，通过对本篇的学习，可使用户掌握 UNIX 和 Linux 系统的常用管理内容。

编程与开发篇：包括 bshell 编程、C/C++及其他编程环境两章。通过对本篇的学习，可为系统综合管理和开发打下基础，从而使用户的管理、应用与开发水平提升到一个新层次。

网络服务与应用篇：包括网络时间服务 NTP、基础服务 SSH 和 VNC、Web 服务器 Apache、DHCP 服务器、FTP 与 TFTP 服务器、网络资源共享服务 Samba 和 NFS、域名服务器 DNS 等，重点介绍的是 Linux 的网络服务及应用。通过对本篇的学习，可使用户胜任网络应用、网络服务配置与管理等工作。

本书特点

本书的组织与编写基于编者二十多年的 UNIX/Linux 管理、应用和开发经验，力求层次

清楚、概念清晰、内容翔实、可操作性强，既便于读者循序渐进地系统学习，又能够使读者了解到 Linux 新的进展，具有以下特点：

（1）从“应用型”出发，兼顾基本知识和基础理论介绍，具有较强的实用性和指导性。

（2）基于红帽（Fedora 24、CentOS 7 和 RHEL 7.3）和 Ubuntu 16.04 系统，具有广泛的代表性和实用性。

（3）涵盖了 UNIX/Linux 系统管理和应用的几乎所有内容。

（4）重点突出实例和操作步骤。

（5）强调系统安全，强化对防火墙和 SELinux 的支持。

（6）每章后都提供一定数量的习题和实验内容。

（7）提供有电子课件、习题解答和实验指导。

适用对象

本书作为“高等教育计算机学科‘应用型’教材”系列之一，可适用于大专院校 UNIX 和 Linux 操作系统教材，也可作为网络操作系统的实例教材，还可作为 UNIX 和 Linux 系统管理者的参考书，更是 UNIX 和 Linux 系统爱好者的益友。

编者信息

本书由邵国金担任主编，褚龙现和张娜担任副主编，郭玉东教授主审。参编人员还有陈红军、董哲、沙锋、苏靖枫和蔡照鹏。

在本书的编写过程中，参考了大量的专业书籍、互联网信息和 Fedora、CentOS、RHEL 及 Ubuntu 系统在线文档、网站信息及相关其他文档，不能一一列出，在此一并表示感谢，特别对郭玉东教授的辛勤工作和大力支持表示衷心的感谢。

面对 Linux 的迅速更新和发展，编者对完成一个高标准的编写任务感到压力很大，限于水平和经验，加之时间仓促，疏忽之处难免，欢迎广大专家、读者批评指正。作者衷心地希望能得到大家，尤其是广大同学和老师的支持与帮助，共同交流 Linux 系统的课程教学经验，探讨 Linux 课程的教学内容和方法，从而提高教学水平，进而提高 Linux 系统的管理和应用水平。作者邮箱：pdsshaogj@hncj.edu.cn。

编　者

2017 年 10 月

改版说明

Linux 是一个变化的世界，一个发展的世界，自从她出生后就一直在不断发展、变化、革新，这正是她的魅力所在，也因此赢得了无数黑客和计算机爱好者们的青睐，并为之团结奋斗。同时，也是二者相互促进，才使 Linux 得以健康、快速发展，且在其发展过程中不断博采众长，引进和使用新技术。移动开发、虚拟化、云计算机和大数据技术的出现及在 Linux 系统的实现与应用，更为 Linux 的发展提供了广阔的施展空间，Hadoop、Openstack、天河二号、神威·太湖之光等都有 Linux 的身影。

本书的第 1 版是基于 Red Hat Linux 9 的，第 2 版是基于 Fedora 9 的，这次改版还是沿着这条线路，是基于 Fedora24 的，但是考虑到实际需要，内容又涵盖了 CentOS 7 和 RHEL 7.3 及 Ubuntu 16.04。Fedora 的发展经历了很多年，也经历了很多代，从 2003 年的 Fedora Core 1，到 2006 年 Fedora Core 6 终结，2007 年更名并发布 Fedora 7，到本书第 2 版定稿时的 Fedora 16，再到本书使用的 Fedora 24，每次变化都会增加新技术和内容，最后被认可的技术会加入 RHEL 和 CentOS 中。实际上，到本书定稿时，Fedora 已经发展到了 Fedora 26，Ubuntu 发展到 Ubuntu 17.04，同时 CentOS 和 RHEL 也都不断更新，真是令人感到日新月异，尤其是其桌面系统的变化更是让人目不暇接，有时也让用户有疲于奔命之感，但无论怎样，发展总是好的，值得我们跟下去。这些都是本书改版的动力。

本次改版使用的 Fedora 24、CentOS 7、RHEL 7.3 和 Ubuntu 16.04 系统是当时最新、最稳定的版本，由于发展速度太快，没办法跟上最新发布版本，但有一点可以肯定，不管 Linux 发展有多快，作为用户使用的基本部分并没有太大变化。首先，对于一般用户来讲，除了图形界面外，影响不大，且图形界面部分也不是本书侧重的内容。其次，作为大学里的教学内容，强调的是基础应用和基本操作，也没有必要追赶最新系统。另外，还有一个因素，那就是时间，编者也不能在很短的时间内针对最新系统的内容做出快速反应。但从整体上讲，本书既考虑到了对基本知识的学习，也考虑到了最新知识和技术的引入。

本次改版所涉及的内容不少，与第 2 版相比，尽管结构没有大的变化，仍分为 4 篇，但几乎所有地方都有所改动。增加了 Ubuntu 16.04 和红帽家族 CentOS 7、RHEL 7.3 的内容，对所有与图形界面相关的部分进行了重写，调整了部分章节的顺序，将第 2 版第 16 章 Linux 系统的安全调整到了管理部分，也重写了第 4 篇，删除了 squid 代理服务器，增加了网络时间服务器 NTP 和 VNC 的内容。第 1 章改动不大，只增加了虚拟化、云计算和大数据等的简介；第 2 章重写了涉及图形界面的部分；第 3 章调整了部分内容顺序，移走了 shell 启动，增强了时间管理和进程管理内容，增加了信号处理内容；第 4 章增加了 sudo 和 write；第 5 章相对稳定，移走了 file 命令，增加了 locate 命令；第 6 章删除了进程状态及转换，移走了 sudo，增加了系统的 crontab；第 7 章增加了对 GPT 格式磁盘的介绍、增加了对系统启动与服务管理包 systemd 的介绍，增加了对 Ubuntu 系统的软件包和服务管理内容的介绍，增加了对 Ubuntu 系统的升级管理部分内容的介绍；第 8 章删除了 d_bus 等内容，增加了 udev 及设备自动发现和设备文件自动分配的内容，增加了系统设备查询命令；第 9 章增加了网络服务管理工具

NetworkManager，增加了 ip 命令，移走了服务管理部分内容；第 10 章为第 2 版第 16 章 Linux 系统安全，增加了 Firewalld 和 ufw 防火墙内容；第 11 章为第 2 版第 10 章 shell 编程，改动不大，但引入了更多的 bash 内容，更名为 bshell 编程；第 12 章为第 2 版第 11 章，现更名为“C/C++及其他编程环境”，增加了对 Objective-C、Python、R 语言及虚拟化、云计算、大数据应用 Openstack 和 Hadoop 的介绍；第 13 章网络时间服务 NTP 为新增内容，介绍了 ntp 和 chrony 两个 NTP 协议的实现及应用；第 14 章为基础服务 OpenSSH 和 VNC，VNC 为新增内容；第 15 章为第 2 版的最后一章，除进行了顺序调整外，也都针对新系统进行了重写，并增加了各服务与防火墙和 SELinux 的关系。

本书有如下几点约定：

（1）本书所用系统为 Fedora 24（工作站）、CentOS 7、RHEL Server 7.3（GUI）、Ubuntu 16.04（桌面），书中所说的红帽系统指的是 Fedora 24、CentOS 7 或 RHEL Server 7.3，Ubuntu 指的是 Ubuntu 16.04。

（2）所用系统均有桌面环境，目的是提供一个友好、多窗口、多“终端”的交互界面，但由于桌面风格不同，书中描述问题时尽量回避桌面环境。

（3）由于涵盖了 Fedora 24、CentOS 7、RHEL 7.3 和 Ubuntu 16.04 系统，对问题的描述尽最大可能采用各系统通用者，如果不特殊声明，则是针对所有系统通用的。比如，到 16.04，Ubuntu 提供了红帽早已有且可以通用的系统启动与服务管理的统一软件包 systemd，所以各系统的启动和服务管理都使用 systemctl 命令，不再考虑个性，但也有无法统一者，如软件包管理，只能分述，红帽系统使用 yum（或 dnf），Ubuntu 系统使用 apt。

（4）由于截止到 16.04，Ubuntu 默认还没有启用 SELinux，本书遵守这个默认，所有与 SELinux 相关的内容与 Ubuntu 无关，只适用于红帽系统。

（5）Ubuntu 系统默认是不启用 root 用户的，但为了描述方便，在执行具有管理功能的命令时，很少或没有使用 sudo 命令，而启用了 root 用户，以 root 用户的身份直接执行。关于这点，敬请 Ubuntu 系统谅解。

（6）本书中独占一行的命令之前的“#”或“$”分别是 root 或普通用户的提示符。一般来说，“#”引导的是只能以 root 用户身份执行的具有管理功能的命令，而“$”引导的是以普通用户身份执行的，但有时控制的并不严格。事实上，如果配置正确，所有具有管理功能的命令均可以普通用户身份使用 sudo 命令来执行。

（7）关于 SELinux 和防火墙，为了方便，在很多书籍或网络资料中都建议关掉它们。本书不建议这样做，因为安全问题不可回避，关键是要正确设置。防火墙软件包有 Firewalld 和 ufw，用户可选择使用。

目　　录

基础篇

第 1 章　系统简介……2
1.1　UNIX 系统简介……2
1.1.1　UNIX 系统的发展历史……2
1.1.2　UNIX 系统的特点……5
1.2　Linux 系统简介……7
1.2.1　Linux 系统的发展历史……7
1.2.2　Linux 系统的特点……8
1.2.3　Linux 系统的发行版本介绍……9
1.2.4　Linux 系统的应用……11
习题 1……12
第 2 章　系统入门……13
2.1　系统的开机与界面切换……13
2.1.1　系统的开机……13
2.1.2　两种操作界面及切换……14
2.2　用户的登录与注销……15
2.2.1　系统的登录……15
2.2.2　注销……17
2.2.3　CLI 终端仿真界面……18
2.2.4　程序的启动……18
2.3　Linux 系统的关闭与重启动……19
2.3.1　字符界面……19
2.3.2　系统的运行级别及切换……20
2.3.3　图形界面……21
2.4　Linux 系统的图形界面介绍……21
2.4.1　GNOME 桌面简介……22
2.4.2　Ubuntu 的 Unity 桌面简介……24
2.4.3　设置中心（Settings）……25
2.5　Linux 系统的在线帮助与资源……27
2.5.1　man……27
2.5.2　textinfo……29
2.5.3　yelp……29
2.5.4　Linux 系统的其他帮助和资源……29
2.6　LibreOffice 办公套件简介……30
2.6.1　LibreOffice Writer……31
2.6.2　LibreOffice Calc……31
2.6.3　LibreOffice Impress……32
2.6.4　LibreOffice Draw……32
习题 2……32
实验 2……33
第 3 章　shell 与 shell 基本操作……34
3.1　shell 基本功能与基本概念……34
3.1.1　shell 基本功能……34
3.1.2　字符与保留字……34
3.1.3　文件命名及文件类型……36
3.1.4　目录结构与路径……38
3.1.5　shell 命令格式与命令编辑……40
3.1.6　标准流与输入/输出重定向……42
3.1.7　管道与三通……44
3.1.8　环境变量与变量……45
3.1.9　引号机制、命令替换与变量替换……46
3.1.10　shell 命令的执行……47
3.1.11　shell 种类……48
3.2　Linux 系统的基本命令……49
3.2.1　目录基本操作命令……49
3.2.2　文件操作基本命令……51
3.2.3　文件属性基本操作……57
3.2.4　文本文件编辑与操作基本命令……59
3.2.5　日期、时间与时区管理命令……67
3.2.6　进程管理基本命令……70
3.2.7　文件或目录比较命令……75
3.2.8　其他操作命令……79
习题 3……85
实验 3……86

系统管理篇

第 4 章　用户、组和密码管理……88
4.1　UNIX 系统的用户和组……88
4.1.1　用户与 uid……88

4.1.2 组与 gid……88
4.2 与用户和组管理相关的文件……88
4.2.1 /etc/passwd……88
4.2.2 /etc/shadow……89
4.2.3 /etc/group……90
4.2.4 /etc/login.defs……90
4.2.5 其他文件……90
4.3 用户管理命令……91
4.3.1 增加用户（useradd）……91
4.3.2 用户删除（userdel）……93
4.3.3 用户修改（usermod）……93
4.4 组管理命令……93
4.4.1 组创建（groupadd）……94
4.4.2 组删除（groupdel）……94
4.4.3 组修改（groupmod）……94
4.5 密码管理……95
4.5.1 密码管理综述……95
4.5.2 密码管理命令（passwd）……95
4.5.3 密码管理示例……96
4.6 用户、组和密码管理图形界面……96
4.6.1 添加用户……97
4.6.2 用户属性修改……97
4.6.3 用户删除……98
4.7 与用户身份和位置相关的其他命令……98
4.7.1 显示与用户和组相关的身份信息（id）……98
4.7.2 显示已登录用户的信息（who）……99
4.7.3 显示使用者的用户名（whoami）……100
4.7.4 向系统中的指定用户发信息（write）……100
4.7.5 向系统中已登录的所有用户发信息（wall）……100
4.7.6 确定用户所使用的终端设备（tty）……100
4.7.7 不退出系统而将自己切换成其他用户（su）……101
4.7.8 以其他用户身份执行程序（sudo）……102
习题 4……104
实验 4……104
第 5 章 文件系统管理及使用……105
5.1 文件系统权限及管理……105
5.1.1 三种权限……105
5.1.2 三类人……105
5.1.3 权限控制……105
5.1.4 默认权限与 umask……106
5.2 权限管理命令……107
5.2.1 设置文件创建掩码（umask）……107
5.2.2 改变文件的权限（chmod）……108
5.2.3 改变文件的所有者（chown）……108
5.2.4 改变文件的组（chgrp）……109
5.2.5 ext2+文件系统的新增属性及其管理……109
5.3 文件系统管理……111
5.3.1 UNIX/Linux 支持的文件系统……111
5.3.2 UNIX/Linux 系统使用的存储设备……113
5.3.3 磁盘分区管理与文件系统的创建……114
5.3.4 文件系统的使用……119
5.3.5 文件系统的检查、修复与同步……124
5.4 与文件系统管理相关的其他命令……126
5.4.1 文件综合查找命令（find）……126
5.4.2 文件按名查找命令（locate）……128
5.4.3 文件复制命令（dd）……129
5.4.4 链接管理命令（ln）……130
5.4.5 特别文件创建（mknod，mkfifo）……130
5.4.6 磁盘空间和文件系统的使用情况统计（df）……131
5.4.7 目录使用磁盘空间情况统计（du）……132

5.4.8 数据备份与文件归档管理（tar，cpio）…132
5.4.9 文件的压缩与解压缩…136
5.5 图形界面下的文件和目录管理…139
习题 5…140
实验 5…141
第 6 章 进程、任务与作业管理…142
6.1 程序和进程的概念…142
6.1.1 程序、进程、作业和任务…142
6.1.2 调度策略与优先级的计算…142
6.1.3 Linux 操作系统的启动…142
6.1.4 0#进程与 1#进程…143
6.1.5 进程状态及查询（ps）…144
6.1.6 三类进程…145
6.2 登录 shell 的启动与定制…146
6.2.1 用户登录过程与登录 shell 的启动…146
6.2.2 用户登录控制与 shell 的定制…147
6.3 Linux 系统的启动过程分析…148
6.3.1 SysVinit…148
6.3.2 upstart…151
6.3.3 systemd…154
6.4 服务管理…158
6.4.1 systemctl…158
6.4.2 早期的服务管理…159
6.4.3 超级服务器（xinetd）…161
6.4.4 rc-local.service…164
6.5 进程管理与调度命令…164
6.5.1 可执行文件的 setuid、setgid 权限和目录的 sticky 属性…164
6.5.2 进程管理与调度命令…166
6.5.3 改变进程的家目录（chroot）…170
6.6 任务的自动调度…172
6.6.1 at 和 batch…172
6.6.2 crontab…174
6.6.3 系统 crontab…176
6.7 进程管理图形界面…176
习题 6…176
实验 6…177
第 7 章 系统安装、扩充、启动与管理…178
7.1 系统安装…178
7.1.1 安装任务与准备…178
7.1.2 硬盘的物理结构与分区划分…178
7.1.3 安装 Linux 系统所需的基本分区…182
7.1.4 安装过程…183
7.1.5 虚拟机的安装与使用…187
7.2 引导器（grub）…188
7.2.1 grub 常用术语…188
7.2.2 操作界面…189
7.2.3 grub 及其配置…190
7.2.4 系统的启动及启动参数修改…195
7.3 软件包管理…197
7.3.1 红帽系统的软件包管理…197
7.3.2 ubuntu 的软件包管理…201
7.3.3 其他格式的软件包管理…204
7.3.4 图形界面…206
7.4 系统升级…206
7.5 日志管理…207
7.5.1 日志系统…207
7.5.2 常见日志文件及阅读…209
7.5.3 日志滚动…211
7.6 系统管理…212
7.6.1 系统管理的任务…212
7.6.2 系统管理的工具与命令…213
7.7 内核配置与参数在线调整…218
7.7.1 内核配置、编译与安装…219
7.7.2 编译与安装新内核…221
7.7.3 模块管理…222
7.7.4 内核参数在线调整…223
习题 7…223
实验 7…224
第 8 章 设备管理…225
8.1 设备管理概述…225
8.1.1 Linux 系统支持的设备…225
8.1.2 设备发现与 udev 简介…226
8.1.3 系统设置与查看…227
8.2 打印机的管理与使用…229

8.2.1 cups 的安装……230
8.2.2 打印机的安装与配置……230
8.2.3 cups 系统的启动管理……233
8.2.4 打印机使用……234
8.2.5 打印任务管理……235
8.2.6 cups 的配置文件……236
8.3 交换区管理……236
8.3.1 概述……236
8.3.2 使用交换设备……237
8.3.3 使用交换文件……238
8.4 串口的管理与使用……238
8.4.1 Linux 系统的串口设备……238
8.4.2 串口管理程序（setserial）……239
8.4.3 串口和调制解调器管理及通信程序（minicom）……240
习题 8……242
实验 8……243
第 9 章 网络配置、管理与基本应用……244
9.1 TCP/IP 基础知识……244
9.1.1 IP 地址……244
9.1.2 端口及服务……246
9.1.3 物理地址与逻辑地址……247
9.1.4 主机名及设置……247
9.1.5 网卡命名方案……248
9.2 TCP/IP 配置……249
9.2.1 与网络有关的配置文件……249
9.2.2 网络服务及管理……254
9.2.3 IP 地址配置……255
9.3 网络管理命令……258
9.3.1 测试网络是否通（ping）……258
9.3.2 检查网络状态（netstat）……259
9.3.3 地址解析协议缓存中的项目管理（arp）……261
9.3.4 网络接口配置（ifconfig）……262
9.3.5 网络 IP 管理（ip）……263
9.3.6 网络接口的启用与停止（ifup/ifdown）……264
9.3.7 路由表维护（route）……264
9.3.8 主机及地址查询（nslookup/host）……266
9.3.9 网络路由跟踪（traceroute）……267
9.4 网络应用常用命令简介……267
9.4.1 远程登录……267
9.4.2 文件传输……268
9.4.3 邮件收发……268
9.4.4 文件下载……270
习题 9……270
实验 9……271
第 10 章 Linux 系统的安全……272
10.1 Linux 系统安全概述……272
10.1.1 Linux 操作系统的基本安全机制……272
10.1.2 Linux 系统可能遇到的安全问题及防范策略……273
10.2 检查和监督系统的运行情况……276
10.2.1 检查网络……276
10.2.2 用 ps 或 pstree 检查进程……276
10.2.3 检查系统的日志文件……276
10.2.4 停止不需要的服务……276
10.2.5 去掉多余的具有 SUID 和 SGID 属性的文件……277
10.3 入侵检测和事件报告……277
10.3.1 使用完整性检查工具……277
10.3.2 事件报告制度……278
10.4 防火墙……279
10.4.1 iptables……279
10.4.2 Firewalld……281
10.4.3 ufw……285
10.5 SELinux……286
10.5.1 SELinux 中的安全类型和角色……286
10.5.2 SELinux 中的策略……288
10.5.3 SELinux 的 Targeted 策略……288
10.5.4 SELinux 的策略及改变……289
10.5.5 SELinux 中的布尔值及改变……290
10.5.6 检查 SELinux 的状态……290
10.5.7 SELinux 安全上下文管理……291

10.5.8 SELinux 的应用 ……………… 293
习题 10 ……………………………………………… 295
实验 10 ……………………………………………… 295

编程与开发篇

第 11 章 bshell 编程 ………………………………… 297
11.1 正则表达式 ………………………………… 297
11.1.1 字符集 ………………………………… 297
11.1.2 shell 正则表达式 ……………… 299
11.2 流编辑（sed） ………………………… 299
11.2.1 功能及用法 ……………………… 299
11.2.2 参数与说明 ……………………… 300
11.2.3 脚本命令 ………………………… 300
11.2.4 sed 示例 ………………………… 301
11.3 模式搜索与处理（awk） …………… 302
11.3.1 功能及用法 ……………………… 302
11.3.2 参数说明 ………………………… 302
11.3.3 记录和域 ………………………… 302
11.3.4 变量 ……………………………… 302
11.3.5 操作符 …………………………… 303
11.3.6 控制语句 ………………………… 304
11.3.7 常用函数 ………………………… 304
11.3.8 awk 程序的执行 ………………… 305
11.3.9 awk 使用示例 …………………… 305
11.4 Bourne shell 及其编程 ………………… 306
11.4.1 特殊字符 ………………………… 306
11.4.2 I/O 重定向 ……………………… 307
11.4.3 变量与参数 ……………………… 308
11.4.4 shell 的状态 ……………………… 309
11.4.5 shell 的调用与变量传递 ……… 310
11.4.6 shell 程序设计 ………………… 311
11.4.7 命令行参数与选项的处理 ………………………………… 324
11.4.8 shell 程序调试 ………………… 328
11.4.9 shell 脚本程序格式 …………… 329
习题 11 ……………………………………………… 332
实验 11 ……………………………………………… 333
第 12 章 C/C++及其他编程环境 ……………… 335
12.1 编译器 ……………………………………… 335
12.1.1 功能及用法 ……………………… 335
12.1.2 参数及说明 ……………………… 336
12.1.3 示例 ……………………………… 336
12.1.4 gcc/g++的工作过程 …………… 338
12.2 头文件 ……………………………………… 338
12.3 链接器与库文件 ………………………… 339
12.4 静态库 ……………………………………… 339
12.4.1 静态库的管理 …………………… 339
12.4.2 构造静态库 ……………………… 340
12.4.3 使用自己的库 …………………… 340
12.5 共享库 ……………………………………… 341
12.5.1 共享库构造 ……………………… 341
12.5.2 共享库的管理 …………………… 341
12.5.3 共享库编程 ……………………… 342
12.5.4 共享库使用示例 ………………… 343
12.6 make 与 Makefile ………………………… 344
12.6.1 make 的用法简介 ………………… 344
12.6.2 Makefile 文件 …………………… 345
12.6.3 Makefile 示例 …………………… 347
12.7 调试器（gdb） …………………………… 350
12.7.1 gdb 功能 ………………………… 350
12.7.2 gdb 基本命令 …………………… 351
12.7.3 程序调试方法 …………………… 351
12.8 UNIX/Linux 其他编程工具简介 ……353
12.8.1 常用库与 GNOME/GTK 开发 ………………………………… 353
12.8.2 KDevelop/Qt 开发 ……………… 354
12.8.3 eclipse …………………………… 354
12.8.4 Java 开发 ………………………… 355
12.8.5 Perl 开发 ………………………… 356
12.8.6 数据库开发 ……………………… 357
12.8.7 PHP 开发 ………………………… 358
12.8.8 Objective-C 开发 ………………… 358
12.8.9 Python …………………………… 359
12.8.10 R 语言 ………………………… 360
12.8.11 虚拟化、云计算和大数据应用 ……………………… 363
习题 12 ……………………………………………… 364
实验 12 ……………………………………………… 365

网络服务与应用篇

第 13 章 网络时间服务 NTP …………………… 367
13.1 NTP 协议 ………………………………… 367

13.1.1 NTP 简介 367
13.1.2 NTP 的工作模式 367
13.1.3 NTP 服务器选择 368
13.2 通过 ntp 配置 NTP 368
13.2.1 软件包的安装 368
13.2.2 服务管理 368
13.2.3 配置文件及配置 369
13.2.4 文件/etc/sysconfig/ntpd 和 /etc/default/ntp 371
13.2.5 防火墙设置 371
13.2.6 配置文件实例 371
13.2.7 配置测试 372
13.2.8 无 NTP 服务的 NTP 客户端 374
13.3 通过 chrony 配置 NTP 375
13.3.1 chrony 软件包的安装 375
13.3.2 服务管理 375
13.3.3 配置文件 chrony.conf 及默认配置 376
13.3.4 配置实例 377
13.3.5 配置测试 378
13.3.6 其他说明 380
习题 13 380
实验 13 380
第 14 章 基础服务 OpenSSH 和 VNC 381
14.1 OpenSSH 381
14.1.1 OpenSSH 协议 381
14.1.2 软件安装与启动管理 381
14.1.3 配置文件与服务器设置 382
14.1.4 客户端及应用 385
14.1.5 与防火墙和 SELinux 的关系 387
14.2 VNC 388
14.2.1 红帽的 TigerVNC 388
14.2.2 客户端及使用 390
14.2.3 Ubuntu 的 x11vnc 390
习题 14 391
实验 14 391
第 15 章 Web 服务器 Apache 392
15.1 Apache 概述 392
15.2 Apache 的安装和启动 392
15.2.1 Apache 的安装 392
15.2.2 Apache 服务器启动管理 393
15.2.3 Apache 服务的测试 394
15.3 Apache 的配置文件和配置指令 394
15.3.1 Apache 主配置文件的结构 394
15.3.2 Apache 的配置指令 395
15.4 Web 服务器配置实例 404
15.4.1 Web 服务器配置实例 405
15.4.2 Fedora 的图形配置工具 409
15.5 httpd 与防火墙和 SELinux 的关系 409
15.5.1 httpd 与防火墙的关系 410
15.5.2 httpd 与 SELinux 的关系 410
习题 15 411
实验 15 411
第 16 章 DHCP 服务器 412
16.1 DHCP 介绍 412
16.1.1 DHCP 协议 412
16.1.2 DHCP 的工作过程 413
16.2 DHCP 服务器的安装与启动管理 414
16.2.1 DHCP 服务器软件安装 414
16.2.2 启动管理 414
16.3 DHCP 的配置 415
16.3.1 配置文件 415
16.3.2 配置文件中的定义、参数和选项及意义 417
16.4 DHCP 规划 419
16.4.1 在不同的网络中使用 DHCP 419
16.4.2 设置 DHCP 中继 419
16.4.3 设置备份 DHCP 420
16.5 DHCP 客户端设置 421
16.6 DHCP 与防火墙的关系 421
习题 16 422
实验 16 422
第 17 章 FTP 与 TFTP 服务器 423
17.1 FTP 与 FTP 服务器概述 423
17.1.1 FTP 的相关概念 423
17.1.2 Linux 系统的 FTP 服务器 425

17.2 vsftpd 服务器 ………426
17.2.1 vsftpd 服务器的安装与启动 ………426
17.2.2 vsftpd 的配置………427
17.2.3 vsftpd.conf 的常见应用配置 ………432
17.3 FTP 服务器的使用………435
17.3.1 用浏览器访问………435
17.3.2 使用客户端命令 ftp ………435
17.4 TFTP 与 TFTP 服务器的使用简介………439
17.4.1 TFTP 协议 ………439
17.4.2 TFTP 的安装和配置及应用 ………439
17.5 与防火墙和 SELinux 的关系 ………441
17.5.1 与防火墙的关系………441
17.5.2 与 SELinux 的关系………442
习题 17 ………442
实验 17 ………443
第 18 章 网络资源共享服务………444
18.1 网络资源共享简介………444
18.2 Samba 服务 ………444
18.2.1 SMB 协议与 Samba………444
18.2.2 Samba 的安装与启动管理 ………446
18.2.3 Samba 的配置………447
18.2.4 配置共享打印机………452
18.2.5 Samba 共享服务使用 ………453
18.2.6 Samba 图形界面配置 ………455
18.2.7 关于防火墙和 SELinux 的说明 ………456
18.3 NFS 服务器………457
18.3.1 NFS 介绍………457
18.3.2 NFS 文件系统配置 ………458
18.3.3 NFS 系统的使用 ………461
18.3.4 NFS 的其他功能 ………462
18.3.5 关于 NFS 的其他说明 ………465
习题 18 ………466
实验 18 ………466
第 19 章 域名服务器 DNS………467
19.1 DNS 概述………467
19.1.1 IP 与域名的转换………467
19.1.2 域名空间和区域 ………468
19.1.3 DNS 查询 ………469
19.1.4 客户端与域名解析相关的配置文件 ………470
19.1.5 DNS 服务器的类型………470
19.2 BIND………471
19.2.1 BIND 简介 ………471
19.2.2 安装 BIND ………471
19.2.3 bind 的启动管理 ………472
19.2.4 DNS 服务器的运行方式及工作目录………472
19.2.5 DNS 服务器配置基础………473
19.3 DNS 服务器配置示例 ………479
19.4 测试 DNS 服务器 ………481
19.5 DNS 与防火墙及 SELinux 的关系 ………483
19.5.1 DNS 与防火墙的关系………483
19.5.2 DNS 与 SELinux 的关系………484
习题 19 ………484
实验 19 ………484
参考文献 ………485

基础篇

- 系统简介
- 系统入门
- shell 与 shell 基本操作

第 1 章　系统简介

1.1　UNIX 系统简介

1.1.1　UNIX 系统的发展历史

1．Multics 与星际旅行

在 20 世纪 60 年代，大部分操作系统为批处理系统，交互性差。1965 年，AT&T 贝尔实验室（Bell Labs）、通用电气公司、麻省理工学院 MAC 课题组一起联合为美国国防部研制开发了一个称为 Multics（MULTiplexed Information and Computing System）的新操作系统。Multics 项目的设计规模宏大，可谓“完美”，但是受当时计算机软硬件水平的影响，最终未能完成设计目标。Multics 系统的核心内容是能够很漂亮地支持大群用户对大型计算机的交互式分时使用。当时，用户与计算机交互表现为用户在键盘上输入，计算机通过电传打字机响应（因为当时还没有显示器）。

Multics 项目终止后，当贝尔实验室从 Multics 研究联盟中退出时，Ken Thompson 带着从 Multics 激发的灵感留了下来，写出了一个名叫“星际旅行（Space Travel）”的游戏程序。Ken Thompson 找到了一台废弃的 DEC PDP-7 计算机运行他的程序，这台 PDP-7 成为了“星际旅行”的游戏平台和 Thompson 关于操作系统设计思路的试验场。为了使这台机器运行起来，Thompson 把注意力从游戏转向操作系统。

Ken Thompson 和 Dennis Ritchie 一起着手开发 DEC PDP-7 上的操作环境，为支持游戏开发而在 PDP-7 上编制的实用程序成了 UNIX 的核心。这个初期的操作系统只有和现在的 UNIX 比较之后才能勉强被认出来。它的文件系统很原始，也没有实行现在的标准，没有分时使用能力。UNIX 最初的名字是“UNICS”（Uniplexed Information and Computing System）。1970 年，贝尔实验室的另一位研究员 Brian Kernighan 提出 UNIX 这个名字，UNIX 中的 Uni 与 Multi 相对应，意为没有那么复杂，而 x 则是 cs 的谐音。

当时的计算机软硬件环境给 UNIX 带来了永久性的影响。当时计算机硬件的水平相当原始，最强大的机器所拥有的计算能力和内存还不如现在一个普通手机。所谓的大硬盘容量也不超过 1MB。视频显示终端刚刚起步，6 年以后才得到广泛应用。最早分时系统的标准交互设备就是 ASR-33 电传打字机。因此，只要有可能，UNIX 开发者就使用最短的命令名称和最短的信息。比如，列目录命令 ls，它本来该是 list，只有 4 个字符，但也被简化为了 ls，只剩两个字符。如果命令执行成功了，通常不给出任何信息，而给出一个可被查询的返回码。UNIX 命令简洁、“少说多做”的传统正是从这里开始的。

PDP-7 UNIX 在早期已经拥有现今 UNIX 的诸多共性，它提供的编程环境也比当时读卡式批处理大型机的环境要舒服得多。因此，UNIX 可以称得上是第一个能让程序员直接坐在机器旁，一边编程一边测试的联机系统。

最初的 UNIX 用汇编语言写成，应用程序用汇编语言和解释型语言 B 混合编写。B 语言

小巧、实用，但作为系统编程语言还不够强大，所以 Dennis Ritchie 给它增加了数据类型和结构后，于 1971 年起由 B 语言演化成了 C 语言。

1973 年 11 月，Thompson 和 Ritchie 等人用 C 语言重写了 UNIX，这是 UNIX 操作系统迈向成功之路的关键一步，也成为“可移植操作系统”的开端。有了 C 语言之后，可移植操作系统“已经”变成现实，以至于使得 UNIX 可以被移植到几乎所有的硬件平台。1979 年，Ritchie 评价说：“UNIX 的成功在很大程度上源自其以高级语言作为表述方式所带来的可读性、可改性和可移植性。”

2．UNIX 与黑客文化

与 UNIX 传统的历史交织在一起的有一种隐性文化，一种更难归类的文化，这是一种传达着有关美和优秀设计价值体系的文化，人们把这种文化称为“黑客文化”。这里的黑客（hacker）不是我们现在意义的坏人或捣乱分子，而是水平极高的，热衷于编程和计算机事业的优秀人士。

1974 年，Thompson 和 Ritchie 在《美国计算机通信》（Communications of the ACM）上发表的一篇论文中第一次公开展示了 UNIX。文中作者描述了 UNIX 前所未有的简洁设计，并报告了 600 多例 UNIX 应用，极大地吸引了黑客们和计算机爱好者。

但是，根据 1958 年为解决反托拉斯案例达成的和解协议，AT&T（贝尔实验室的母公司）不允许从事除“公用通信服务”外的任何商业活动，因此 AT&T 被禁止进入计算机相关的商业领域，所以 UNIX 不能够成为一种商品。为了满足黑客和爱好者们的需要，AT&T 公司在签署简单协议的前提下，将 UNIX 系统无偿地提供给大学，以供教学与研究，或通过 Ken Thompson 默默回应请求者，将磁带和磁盘一包包地寄送出去。

UNIX 的发展迅速笼罩着一层反传统文化的氛围，没有版权和费用约束，源代码可以自由交流，UNIX 黑客们沉浸在共同编织未来和编写系统的狂欢中。在当时，大部分 UNIX 程序以源代码形式配送，由最终用户把它编译成可执行程序，黑客不仅使用 UNIX 系统，而且还编写 UNIX 程序，交流、修改和共享源代码。由于众多黑客们的参与，UNIX 得到了快速发展，许多大学都对 UNIX 做出过贡献，多伦多大学计算机系发明了 200dpi 的打印机/绘图仪，并且开发了相关软件；耶鲁大学的计算机专家和学生们改进了 UNIX 的 shell；普渡大学的电子工程系对 UNIX 的性能进行了重要改进，推出了支持大量用户的 UNIX 版本，还推出了最早的 UNIX 网络之一；加州大学伯克利分校的学生开发了新 shell 和许多小型实用工具。

在 UNIX 的发展过程中，加州大学伯克利分校很早就成为最重要的学术热点，尤其 Ken Thompson 于 1975—1976 年休期间在此教学，更对 UNIX 的研究注入了强劲活力。1977 年，伯克利分校毕业生 Bill Joy 管理的实验室发布了第一版 BSD（Berkeley Software Distribution），到 1980 年，伯克利分校成了为这个 UNIX 变种积极做贡献的高校子网的核心，从此 UNIX 走向了以 AT&T 和伯克利分校为主的开发道路，两者相互学习、相互批评，促进了 UNIX 的发展。System V 和 BSD UNIX 成为 UNIX 的两大主流，现在大部分 UNIX 是它们的衍生品。

3．System V 和 BSD UNIX

AT&T 公司与美国司法部的法律大战终于在 1982 年达到终点，AT&T 被重新允许进入计算机市场。贝尔实验室先后在 1983 年发行了 System V 和几种微处理机上的 UNIX，1984 年发行了 System V Release2（SVR2）；1986 年，UNIX System V 发展到了修订版 Res2.1 和 Res3.0；

1987 年发行了 SVR3。System V 引入许多新特征、新设施，最有代表性的就是进程间通信机制（IPC）。

UNIX 系统的另外一个最重要的分支 BSD UNIX 的大部分特点（如 C Shell、vi 编辑器、作业控制、动态安装文件系统和快速文件系统）与 System V UNIX 都差不多，但是促使其迅速流行的原因是加州大学伯克利分校计算机系统研究小组（CSRG）最早发表的一个包括页式内存管理的 UNIX 版本的一系列成就。

FreeBSD 正是 UNIX 众多分支中相当优秀的一支，是著名的 BSD UNIX 的一个继承者，SUN OS 和 Solaris 也是在 DSB 上发展起来的。

4．UNIX 与 TCP/IP

1980 年，国防部高级研究计划局（DARPA，Defense Advanced Research Projects Agency）需要请人在 UNIX 环境下的 VAX 机上实现全新的 TCP/IP 协议栈。伯克利在虚存、开放源代码方面的工作和拥有最强大开发工具的优势促使美国国防部高级计划研究署选择了伯克利 UNIX 作为平台，决定支持伯克利的 UNIX 系统开发。BSD 与美国国防部一起成为将 UNIX 系统实现网络互联的推动力量，该项目促进了 UNIX 的发展，也促进 TCP/IP 的发展。

1983 年夏天，支持 TCP/IP 协议版本的 UNIX—BSD 4.2 公开发行了，这宣布了 TCP/IP 的诞生，具有划时代意义。有了 TCP/IP，一切都变了，造就了 1994 后互联网大爆炸。

在 1983 年 TCP/IP 实现随 Berkeley 4.2 版发布之前，UNIX 对网络的支持一直是薄弱的。早期的以太网实验不尽如人意，贝尔实验室开发了一个难看但还能用的工具 UUCP，可在普通电话线上通过调制解调器来传送软件，可以在分布很广的机器之间转发邮件，支持 Usenet，但是没有 FTP，没有 telnet，只有限制重重的远程作业执行和慢得要死的连接。其他的网络方案也在发展，贝尔实验室和 Sun Microsystems 都发表了网络文件系统，从而把所有网络机器的文件系统结合成一个单独的大文件系统，其中 RFS（Remote File System）和 NFS（Network File System）至今仍在使用。

5．UNIX 向 GUI 发展

图形用户接口（GUI）在 20 世纪 70 年代或更早就开始流传，大部分 GUI 从 Xerox 的 Palo A1to 研究中心的工作中演变而来。GUI 是一个用图形（称为图标）表示操作系统的组成部分，GUI 的最初设想是创建一个友好的用户环境并以一致的隐喻来理解操作系统。GUI 使用方便，不是因为容易理解，而是因为使用的一致性，当用户学会在一种 GUI 环境下运行一个程序后，就可以容易地运行相同环境中的其他程序。

20 世纪 80 年代初，计算机技术的发展使 GUI 产生了一个大飞跃。美国 DEC 公司和麻省理工学院合作的 Athena 计划，合并了一个基于斯坦福大学 W 窗口系统的窗口系统。沿袭 B 语言和 C 语言规矩，新系统命名为 X，或叫 X 窗口系统。

X 很受欢迎，部分原因是其公开的源代码，更重要的是 X 窗口现在可以在很多机型和操作系统上运行，而不仅仅是 UNIX。然而，X 窗口只提供了显示窗口、图形和文本的方法，却没有定义外观。开放软件基金会（OSF）决心建立一个标准的 GUI，该 GUI（称为 Motif）是使用 X 窗口创建的。另一个流行的 GUI 是 Open Look，由 Sun Microsystems 提供。

1985 年，X window 系统的创始人发布了 X window 的源码，而无须版税、约束和授权。这项决策的直接结果就是 X window 成为不同 UNIX 厂商之间合作的安全中立区，成为 UNIX

的图形引擎。

6．UNIX 系统的商业化

UNIX 系统的不断发展吸引了许多计算机公司，它们开始将 UNIX 商业化，出售自己的 UNIX 产品。每一家都以 AT&T 或 BSD 为蓝本，将它们移植到自己的硬件上，并加上一些自己的“增值”功能：Sun Microsystems 公司基于 BSD 开发、发行了 Sun OS，后来又基于 SVR4 发行了 Solaris 系统。微软与 SCO 公司合作发行了 XENIX 系统。SCO 公司将 SVR3 移植到 386 上，并将其命名为 SCO UNIX。IBM 公司开发了 AIX 操作系统。HP 公司开发了 HP-UX。Digital 公司开发了 Digital UNIX。还有著名教授 Andrew S. Tanenbaum 为了教学开发的 MINIX 系统，以及可爱的小企鹅——Linux 操作系统。

每种商业变体都增加了许多新特征，这些功能中的许多后来又被加入到主线系统中，逐步推动 UNIX 系统向前发展。

7．UNIX 系统的标准化

随着 UNIX 的商业化，UNIX 的源代码开始枯竭，各大 UNIX 厂商也都想通过个性发展来谋取优势，各种 UNIX 变种的泛滥造成许多不兼容问题，这使得开发应用的程序员不得不花费大量精力来保证它们的程序可以运行于各种不同的 UNIX 系统中。在这种局面下，美国 IEEE（Institute of Electrical and Electronics Engineers，美国电气和电子工程师学会）组织成立了 POSIX（Portable Operating System Interface of UNIX，可移植操作系统接口）委员会专门从事 UNIX 的标准化工作。

POSIX 标准基于现有的 UNIX 实践和经验，描述了操作系统的系统调用接口，用于保证编制的应用程序可以在源代码一级上、在多种操作系统上移植运行。UNIX 标准意味着一个可以运行 UNIX 应用软件的平台，它为用户提供一个标准的开发界面，而不在于系统内部如何实现。POSIX 委员会完成了 UNIX 系统标准化，各 UNIX 厂商要按其定义重新实现 UNIX。

1.1.2　UNIX 系统的特点

对用户和程序员来说，UNIX 操作系统极富个性，它有许多其他实用操作系统所没有的特点。由于 UNIX 的多用户多任务能力、设备无关性及其提供的很多附加工具，有经验的 UNIX 用户往往比其他系统的用户有更高的收益。UNIX 的错误处理可以在程序运行出错时减小数据损失。UNIX 使用方便，为用户、管理人员、顾客及商业决策人员提供了更大的选择范围。

1．多任务

UNIX 是一个多任务操作系统，即可以同时运行多个任务。通过一个时间分享处理程序，UNIX 实现了多任务。表面上看起来计算机同时为多个程序服务，实际上是在多个进程间很快切换，由于这些处理发生很快以至用户通常察觉不到。

2．多用户

除了多任务能力，UNIX 还可以同时为多个用户服务。UNIX 的多用户主要体现在允许很多用户同时在不同的终端工作，使用操作系统执行相关或不相关的任务，就像每个终端用

户自己都拥有一台单独的机器一样。

终端和 PC 很相似，仅有一个键盘和一个显示器，并用一根电缆连到主机上，但一般来说它不包含处理器。这种终端被人们更形象地称为哑终端。终端仅能接收键盘输入，并再发送给主机，以及接收主机送来的字符，再将其显示到显示器上。某些终端也包含处理器用于辅助用户和主机之间转递数据，如 X 终端拥有图形处理器可以显示窗口和其他图形。

3．并行处理能力

与其他常见的操作系统相比，UNIX 是最早支持多处理机的操作系统，而且它所支持的处理器数目也一直处于领先水平。比如，Windows NT 能支持 1～4 个 CPU，Windows 2000 最多能支持 16 个 CPU，而 UNIX 在 20 世纪 90 年代中期就已经能支持 32～64 个 CPU。UNIX 可以管理数百个乃至成千上万个 CPU 组成的巨型机系统。

4．设备无关性

UNIX 把设备作为特殊文件来处理，并分别命名，这样在任何使用文件名的地方都可以使用设备名，通过文件操作可以实现对设备的操作，并且屏蔽对设备操作的细节。由于大部分设备除共性外还有其独有的功能，如可以格式化磁盘、回绕磁带、调整终端设置等，UNIX 提供了专门的命令和函数执行这些操作，但是这种灵活性并不意味着可对任何文件或设备任意执行操作，还要考虑设备的特殊性，例如，打印机只能用来输出而不能用于输入。

5．工具

UNIX 在其发展的最初阶段就已采用“工具法”解决问题了。每一个“工具”只完成—个简单任务。通过工具组合，可以完成某个工具无法完成的更大的、更复杂的任务。

为了组合工具，UNIX 提供了管道。管道可以让数据从一个程序流到另一程序，每个程序都可以用某种方式修改数据。管道在 UNIX 命令行中以竖线“|”表示，这看起来确实像一条管道。

工具法在处理更加复杂的程序集合方面有很多优势。

（1）灵活性。可以组合实用的工具来完成最复杂的任务。

（2）速度快。可以快速完成任务，因为基本的工具程序块已经存在。

（3）结构化。UNIX 都使用相同的程序块，鼓励用户使用相似的方式解决问题。

工具法也可能会使 UNIX 的新手感到困惑，因为它可以使用很多种方法进行同一处理，这使得习惯使用图形接口来解决问题的用户就更难体验到工具法的威力了。

6．错误处理

由于 UNIX 是一个多用户系统，所以它必须有很强大的错误处理能力。对于单用户系统，错误引起的不便只是关机和重启动，而多用户系统的宕机和重启动会引起很多用户不便，甚至造成严重后果和经济损失，所以 UNIX 尽力防止危险性的错误也就不足为怪了。

UNIX 防止应用程序不合理地改写内存、防止用户有意或偶然删除不属于自己的文件；还提供了强大的中断处理“越界”、死循环等事件的发生；当出现严重错误时，UNIX 系统保存所有当前状态的信息然后退出，供熟练的管理人员和程序人员查找出错原因。实际上，由于错误处理得当，UNIX 系统极少出现失控的情况。

7．强大的网络功能

UNIX 系统提供了丰富的网络功能，是真正的网络操作系统。作为互联网络技术基础的 TCP/IP 就是在 UNIX 上开发出来的，已经成为 UNIX 系统不可分隔的重要组成部分，也为网络技术的发展可在其他操作系统的应用起到了推动和榜样作用。UNIX 系统还提供有众多常用的通信协议软件和网络服务，通过这些软件或服务，用户也轻松实现自己所需的功能或在不同操作系统间实现互联互通。

8．开放性

UNIX 系统的最本质特征是它的开放性，它是目前开放性最好的操作系统，是目前唯一能稳定运行在从手持机、微机、小型机到中型、大型机的操作系统。所谓开放性是指它遵循统一的国际标准和规范。开放性已成为 20 世纪 90 年代计算机技术的核心问题，也是一个系统或软件是否具有生命力的重要因素。

1.2 Linux 系统简介

Linux 是一个功能强大的操作系统，同时它是一个自由软件，是免费、源代码开放、可以自由使用的 UNIX 兼容产品。它诞生于 1991 年 10 月 5 日（这是第一次正式向外公布的时间），以后借助于互联网络，经过全世界各地计算机爱好者的共同努力，已成为世界上使用最多的一种 UNIX 类操作系统，并且使用人数还在迅猛增长。

1.2.1 Linux 系统的发展历史

Linux 操作系统的诞生、发展和成长过程始终依赖着五个重要支柱：UNIX 操作系统、MINIX 操作系统、GNU 计划、POSIX 标准和互联网络。

1．MINIX 操作系统

MINIX 操作系统是由 Andrew S. Tanenbaum 开发的，主要用于学生学习操作系统原理的实验室用操作系统，到 1991 年时版本是 1.5。目前主要有两个版本在使用：1.5 版和 2.0 版。当时该操作系统在大学是免费使用的，可以从许多 FTP 上下载。

对于 Linux 系统，Tanenbaum 表示了对其开发者 Linus 的称赞，但他认为 Linux 的发展有很大原因是因为他为了保持 MINIX 的小型化，让学生在一个学期内就能学完，而没有接纳全世界许多人对 MINIX 的扩展要求。

作为一个操作系统，MINIX 并不是优秀者，但它同时提供了用 C 语言和汇编语言写的系统源代码。这是第一次使得有抱负的程序员或 hacker 能够阅读操作系统的源代码，在当时这种源代码是软件商一直小心守护着的。

2．GNU 计划

GNU 计划和自由软件基金会（Free Software Foundation，FSF）是由 Richard M. Stallman 于 1984 年创办的，旨在开发一个类似 UNIX 并且是自由软件的完整操作系统：GNU 系统。

为了保证 GNU 软件可自由地“使用、复制、修复、修改和发布”，所有 GNU 软件必须遵守 GNU 的通用公共许可证 GPL（GNU General Public License，GNU GPL）。

GNU GPL 创造性提出了“反版权”（Copyleft），这是一个不同于商业软件“版权所有”（Copyright）的法律概念，它不否认版权，也不反对发布软件时收取费用或取得利益。它的核心是必须把发布者的一切权利给予接受者，必须保证接受者能同时或通过其他渠道得到源程序，并将 GNU GPL 条款附加到软件的版权声明中，使接受者知道自己的权利。GNU GPL 本身也是受法律保护的版权声明。

现在各种使用 Linux 作为核心的 GNU 操作系统正在被广泛使用，虽然它们通常被称作“Linux”，但是严格地说，它们应该被称为 GNU/Linux 系统。

到 20 世纪 90 年代初，GNU 项目已经开发出许多高质量的免费软件，这些软件为 Linux 操作系统的开发创造了一个合适的环境，是 Linux 能够诞生的基础之一。

3．Linux 操作系统的诞生

1981 年，IBM 公司推出享誉全球的微型计算机 IBM PC。在 1981—1991 年间，MS-DOS 操作系统一直是微型计算机上操作系统的主宰。此时计算机硬件价格虽然逐年下降，但软件价格仍然居高不下。当时 Apple 的 Mactonish 操作系统可以说是性能最好的，但是其天价没人能够轻易靠近。

操作系统的另一个计算机技术阵营是 UNIX 世界。然而，UNIX 大多是商用的，PC 小用户根本就不能靠近它。另外，受到 Bell Labs 的许可可以在大学中用于教学的 UNIX 源代码也一直被小心地守卫着不许公开，于是 MINIX 操作系统出现了，并有一本详细的书描述它的设计实现原理，使得几乎全世界的计算机爱好者都通过这本书来学习操作系统的工作原理，其中也包括 Linux 系统的创始者 Linus Benedict Torvalds。

MINIX 虽然很好，但只是一个用于教学目的的简单操作系统，并不是一个强有力的实用操作系统。到 1991 年，GNU 计划已经开发出了许多工具软件，最受期盼的 GNU C 编译器已经出现，但还没有开发出免费的 GNU 操作系统。

1991 年初，Linus 开始在一台 386SX 兼容微机上学习 MINIX 操作系统。通过学习，他逐渐不能满足于 MINIX 系统的现有性能，开始酝酿开发一个新的操作系统。1991 年 10 月 5 日，Linus 在 comp.os.minix 新闻组上发布消息，正式向外宣布 Linux 内核系统的诞生（Free minix- like kernel sources for 386-AT），这段消息可以称为 Linux 的诞生宣言。

4．Linux 名称的由来

Linux 操作系统刚开始时并没有被称作 Linux，Linus 给他的操作系统取名为 freex，意为 free 与 x 的组合，x 即 UNIX 的 x。在他将新的操作系统上载到 ftp.funet.fi 服务器上时，管理员 Ari Lemke 认为既然是 Linus 的操作系统，就取其谐音 Linux 作为该操作系统的名字吧，于是 Linux 这个名称就开始流传下来。

1.2.2　Linux 系统的特点

Linux 除具有所有 UNIX 系统的功能和特点之外，还具有以下特点。

1．自由软件

Linux 是一款在公用许可证 GPL 保护下的自由软件，用户可以通过网络或其他途径免费获得。正是由于这一点，来自全世界的无数程序员参与了 Linux 的修改、编写工作，程序员可以根据自己的兴趣和灵感对其进行改变，从而使 Linux 系统不断壮大。

2．良好的兼容性

Linux 完全符合 POSIX 标准，可兼容现在主流的 UNIX 系统。在 UNIX 系统下可以执行的程序几乎完全可以在 Linux 上运行，为应用系统从 UNIX 系统向 Linux 系统转移提供了可能。

3．良好的界面

Linux 同时具有字符界面和图形界面。在字符界面用户可以通过键盘输入相应的指令来进行操作。它同时也提供了类似 Windows 图形界面的 X-Windows 系统，用户可以使用鼠标对其进行操作。

4．丰富的网络功能

互联网是在 UNIX 的基础上繁荣起来的，Linux 的网络功能当然不会逊色。在 Linux 中，用户可以轻松实现网页浏览、文件传输、远程登录等网络工作，并且还可以作为网络服务器提供 WWW、FTP、E-Mail 等众多服务。

5．支持多种平台

Linux 可以运行在多种硬件平台上，如具有 x86、680x0、SPARC、Alpha 和 PPC 等处理器的平台。此外 Linux 还是一种嵌入式操作系统，可以运行在手机、掌上电脑、机顶盒或游戏机上。2001 年 1 月发布的 Linux 2.4 版内核已经能够完全支持 Intel 64 位芯片架构，同时 Linux 也支持多处理器技术，使系统性能大大提高。

中国“天河二号”运行的是基于 Linux 的麒麟操作系统，集成 16 000 个运算节点，每节点配备两颗 Xeon E5 12 核心的 CPU，三个 Xeon Phi 57 核心的协处理器，累计 32 000 颗 Xeon E5 主处理器和 48 000 个 Xeon Phi 协处理器，共 312 万个计算核心，这是多么惊人的数字。从 2010 年开始，连续 6 年蝉联世界超级计算机 TOP500 冠军。

1.2.3　Linux 系统的发行版本介绍

Linux 是一个免费的操作系统，用户可以免费获得其源代码，并按照自己的意愿修改。Linux 的开放源代码规则如下。

（1）任何人可以免费使用该操作系统，但不得将其作为商品出售。

（2）任何人可以对该操作系统进行修改，但必须将其修改以源代码形式公开。

（3）如不同意以上规定，任何人无权对其进行复制或从事任何行为。

Linux 是在通用公共许可证 GPL（General Public License）保护下的自由软件，版本有很多种。

1．Linux 的内核版本

简单来说，Linux 的内核就是这个系统的“最小化”，当系统引导的时候被调入内存。外部程序通过调用其中的函数完成操作，所以它是系统的核心。内核版本号的格式是：

主版本号.次版本号.修正号

例如，Linux 2.6.25 主版本号是 2，次版本号是 6，第 25 次修正。

内核版本号有一个规则：次版本号为偶数的是稳定版本，为奇数的是测试版本。所谓稳定版本是指内核的特性已经固定，代码运行稳定、可靠，不增加新的特性，只是修改错误，而测试版本是指相对于上一个稳定版本增加了新的特性，还处于测试中。

2．软件发行的三种形式

软件发行有以下三种形式。

（1）商业软件（Commercial Software）：先购买后使用，典型代表是 Microsoft 的 Windows。

（2）共享软件（Share Software）：先使用后付费，通常不提供源代码，到期未购买仍在继续使用者被认为是侵权。

（3）自由软件（Free Software）：在发布时向用户提供源代码。通常用户通过网络等多种渠道可得到发布版本。如果用户付费，将得到完美的服务和文档。

3．Linux 的发行版本

光有内核还不能构成一个完整的操作系统，于是一些组织或公司将内核与一些应用程序包装起来就构成了一个完整的操作系统，这就是发行套件，不同的公司或组织的发行套件各不相同，版本号也不相同，尽管它们使用了同一版本的内核。现在的发行版本很多，不下百种，如 Ubuntu、SuSE、RedHat、Mandriva、DSL、KNOPPIX、Debian、PCLinuxOS、Slackware、Gentoo、Kubuntu 和红旗等。

其中，RedHat 系列是目前最为活跃、安装简便、最适合初学者的 Linux 发行版，也是目前世界上最流行的 Linux 发行套件之一，自行开发了 RPM 套件管理程序及 X 桌面环境等众多软件并将其源代码回馈给开源社区。RedHat 又称为“红帽 Linux”，包括 RHEL（Red Hat Enterprise Linux，为收费版本）、Fedora Core（由 RedHat 桌面版本发展而来，免费）、CentOS（RHEL 的社区克隆版本，免费）。RHEL 和 CentOS 的稳定性非常好，适合于服务器使用；Fedora 处于不停的发展之中，不断引入新知识、新技术，生命力极强，适用于研究和学习。

国内 Linux 发行版做的相对比较成功的是红旗和中软两个版本，界面美观，安装比较容易，新版本逐渐屏蔽了一些底层操作，适合于新手使用。两个版本都是源于中国科学院软件研究所承担的国家“863”计划的 Linux 项目。

4．Linux 的获得

Linux 是一个自由软件，获得它的最直接的方法就是在互联网上下载，许多站点都提供 Linux 及相关程序，并且绝大部分都是免费的。Linux 的发行版本很多，很难说清楚有多少种，以下列出较流行的 Linux 发行版本的链接，供参考。

Red Hat Linux（RHEL）：http://www.redhat.com
Fedora Linux：http://www.fedoraproject.org
CentOS Linux: http://www.CentOS.org
Debian Linux：http://www.debian.org
Ubuntu Linux：http://www.Ubuntu.com
SuSE Linux：http://www.SuSE.com
红旗 Linux：http://www.redflag-linux.com
中软麒麟：http://www.cs2c.com.cn/

1.2.4　Linux 系统的应用

Linux 的应用十分广泛，主要应用于以下几个方面。

（1）办公自动化。随着 Linux 技术的发展，特别是 X-Window 领域技术的发展，Linux 在界面美观、操作使用方便等方面都有了长足的进步，逐渐为桌面用户所接受。OpenOffice.org 是其用于办公的套件。

（2）Internet、物联网、移动互联网及网络服务。Linux 被广泛用于互联网和内联网，提供有众多的网络应用和网络服务供用户选择使用。在物联网和移动市场上，Linux 一直都是这个领域中占据主导地位的操作系统，如智能手机、电视机、微波炉、电冰箱等大多基于 Linux 系统设计。

（3）数据库。在 Linux 系统中数据库的选择较多，其中 mSQL、MySQL 和 PostgreSQL 是三个最流行的、开源的和基于 SQL 的数据库。除此之外，用户还可使用商业数据库，如 Oracle、Sybase、Informix 和 DB2 等。

（4）虚拟化、大数据、云计算。大数据是建立在云计算平台上，而云计算开源架构底层是基于 Linux 的。例如，最著名的开源云平台 Openstack 底层基于 Linux 操作系统 Ubuntu、CentOS 或 Redhat。Eucalyptus 项目（Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems）是亚马逊弹性云计算的一个开源实现，它与商业服务接口兼容。Eucalyptus 依赖于 Linux 和 Xen 进行操作系统虚拟化。另一个开放源代码项目 Enomalism，同样基于 Linux，同时支持 Xen 和 KVM（Kernel Virtual Machine）。还有开源项目 Hadoop，是由 Apache 基金会开发的分布式系统基础架构，具有高可靠性、高扩展性、高效性、高容错性和低成本等优点，可在 Ubuntu、CentOS 或 RHEL 上实施运行。

（5）应用开发。Linux 支持多种开发语言，如 C、C++、Java、PHP、Perl、Python、R 和 Objective-C，为基于 Linux 系统的多种应用提供支持。

（6）多媒体和娱乐、游戏。Linux 系统提供有游戏，媒体播放器和流媒体服务器等应用程序。实用性嵌入式系统也已经开始走入市场，如机顶盒、PDA、手机等。

（7）图形与图像处理。Linux 系统被广泛用于图形工作站，为图形与图像处理提供保障。如轰动一时的电影《泰坦尼克号》制作所用的 160 台 Alpha 图形工作站中，有 105 台运行的是 Linux 操作系统。Linux 提供关于图像处理、绘图、图像浏览、图标制作和三维模型设计等大量工具可供用户选择和使用。

习题 1

1．思考题

（1）UNIX 的大部分代码是用一种流行的程序设计语言编写的，该语言是什么?
（2）UNIX 系统的特点有哪些？
（3）什么是 Linux？其创始人是谁？
（4）Linux 操作系统的诞生、发展和成长过程始终依赖的重要支柱都有哪些？
（5）简述 Linux 系统的特点。
（6）常见的 Linux 发行版本都有哪些？

2．单项选择题

（1）Linux 最初是基于（　　）操作系统开发出来的。
A．Ms-DOS　　B．Minix　　C．POSIX　　D．Mac OS
（2）关于 Linux 内核版本的说法，以下错误的是（　　）。
A．表示为主版本号.次版本号.修正号　　B．1.2.3 表示稳定的发行版
C．1.3.3 表示稳定的发行版　　D．2.2.5 表示对内核 2.2 的第 5 次修正
（3）Linux 属于（　　）。
A．商业软件　　B．共享软件　　C．自由软件　　D．以上都不对
（4）自由软件的含义是（　　）。
A．用户不需要付费　　B．软件可以自由修改和发布
C．只有软件作者才能向用户收费　　D．软件发行商不能向用户收费

第 2 章　系统入门

2.1　系统的开机与界面切换

2.1.1　系统的开机

在微机或服务器系统中，加电后，Linux 系统开始启动，启动完成后停留在用户登录界面。若是安装有图形界面的工作站或桌面系统，则在启动完成后，停留在图形登录界面（如图 2-1 所示），否则停留在字符界面（如图 2-2 所示），等待用户登录。

关于启动过程的描述与分析，请参见第 6、7 章。



（a）红帽的 GNOME 图形登录界面　　（b）Ubuntu 的 Unity 图形登录界面

图 2-1　图形登录界面

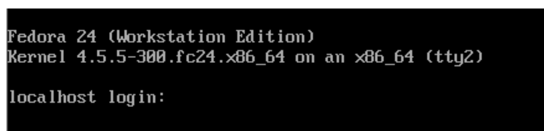
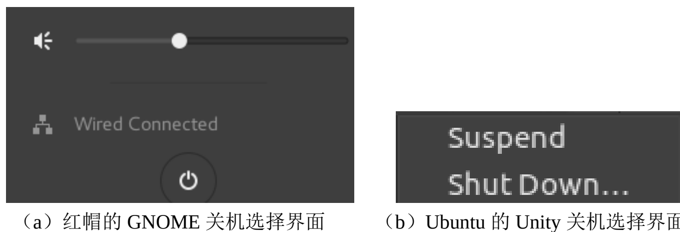


图 2-2　字符登录界面

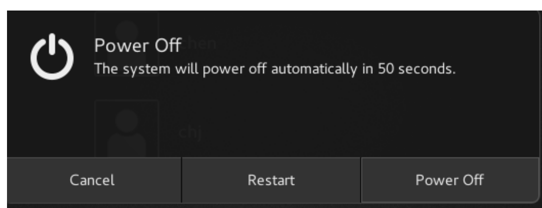
在图 2-1 所示的界面中，若单击右上角的，则可以打开“Universal Access（通用辅助）”设置框界面。通过此界面可以设置的有高对比度（High Contrast）、屏幕键盘（On-screen Keyboard）、屏幕阅读器（Screen Reader）等。用户需要时，可以单击滑块工具的“OFF”或“ON”进行设置。

在图 2-1 所示的界面中，单击右上角最右边的或按钮，可以打开如图 2-3 所示对话框。用户可以单击关机按钮或“Shut Down”弹出如图 2-4 所示的关机对话框，通过此对话框，可允许用户在非登录的情况下选择关闭或重启系统。

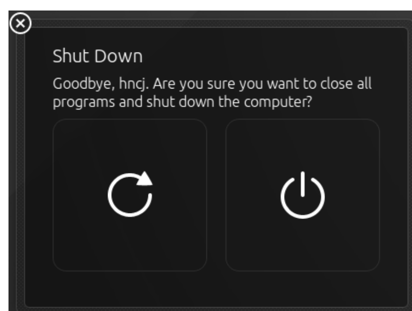


（a）红帽的 GNOME 关机选择界面　　（b）Ubuntu 的 Unity 关机选择界面

图 2-3　系统关闭选择界面



（a）红帽的 GNOME 关机对话框界面



（b）Ubuntu 的 Unity 关机对话界面

图 2-4　系统关闭对话界面

必须注意的是，图 2-1 所示的为一个欢迎登录界面，这是传统 UNIX 和计算机安全学所不提倡的，因为它可能带来安全隐患。

2.1.2　两种操作界面及切换

通常，在桌面或工作站式的 Linux 系统安装过程中，为用户准备了 1 个图形界面（X-Window）和 5 或 6 个字符界面。红帽系统的 5 个字符界面分别位于 F2～F6 上，而图形界面位于 F1 上；Ubuntu 系统的 6 个字符界面分别位于 F1～F6 上，而图形界面位于 F7 上。为了便于记忆，图形界面被称为 GUI（Graphic User Interface），字符界面是基于传统 UNIX 的字符终端操作界面，又被称为 TUI（Terminal User Interface）。默认情况下，在装有图形界面的 Linux 中，启动完毕后会停留在图形界面。用户可以使用组合键在图形界面和字符界面之间切换。

1．从图形界面向字符界面切换

从图形界面向字符界面切换可以通过组合键 Ctrl+Alt+Fm（m=1,2,⋯,6），比如，按下 Ctrl+Alt+F2 可使操作界面切换至 F2 所在的字符界面（参见图 2-2），Ctrl+Alt+F5 可切换到 F5 所在的字符界面。

2．从字符界面向图形界面切换

从字符界面向图形界面切换可以通过组合键 Alt+Fm 来实现（Fm 为图形界面所在的终端号）。

关于图形界面的位置：当只有一个用户使用图形界面登录时，Ubuntu 的登录界面总是在 F7 上；红帽系统的登录界面在 F1 上，登录成功后在什么位置并不确定，要看当时终端的使用情况而定，若开机后首次从桌面系统登录，则登录成功后用户最大可能是工作在 F2 上。另外，当有多个用户同时使用图形界面登录时，登录成功后，工作在不同终端上每个用户的具体位置可以在 TUI 或 CLI 下使用命令 w 或 who 来查询。

需要提醒的是，在有的系统或环境中，从图形界面向字符界面切换时，只使用 Ctrl+Alt+Fm 可能还不够。比如，早期曾使用过 Ctrl+Alt+Fm+Shift；在有的笔记本系统中可能还要使用 Fn 键（Fn 为笔记本中的键名），此时的组合键为 Ctrl+Alt+Fm+Fn。对于从字符界面向图形界面切换也会存在同样问题。

2.2 用户的登录与注销

Linux 系统使用用户身份认证来控制安全问题，并根据用户身份来进行授权管理。用户要使用系统，首要的工作就是登录。登录实际上是用户向系统做自我介绍，又称为身份验证（authentication）。此时需要输入系统已经注册过的用户名和用户密码，如果用户名或密码不正确，将不允许进入系统。用户名也叫账号，密码也称为口令。

在 UNIX/Linux 系统中，不是所有的用户都是与生俱来的，也不是人人平等的。系统安装过程已经创建了一个名为 root 的用户和一些其他用户。root 用户是系统的维护者和管理者，具有至高无上的权力，可以做一切事情，因此在 UNIX/Linux 系统中 root 也叫超级用户或根用户。除 root 之外的所有其他用户都是普通用户，它们只能在 root 分配的职权范围内开展工作，且不得越权行事。普通用户也叫一般用户。

root 或具有管理权限的用户可以通过用户管理命令或工具创建新的用户，并分配其适当权限。UNIX/Linux 的用户使用规则是，除非是系统管理员，或做系统管理与维护工作，否则不要使用 root 身份登录，因为 root 的权力过大，可能因为误操作造成对系统的危害。因此在创建了普通用户账号后，强烈建议使用普通用户账号登录并进行工作。

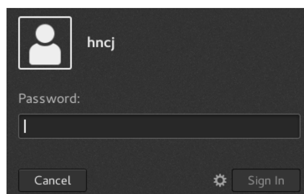
当用户使用完系统或暂离开操作位置时，应该即时注销，需要时重新登录，这样可以避免别有用心的人见缝插针地越权操作。在很多有安全要求的系统中，这既是安全的需要，也是制度的需要，尤其在安全级别要求较高的部门更是如此，但是用户注销不等于关机或关闭系统。注销是用户自己的事情，只是使用户自己退出系统或从系统签退，系统中的其他用户仍可继续工作，但关闭系统却会使得所有用户均不能工作了。在 UNIX/Linux 系统中，关机是系统管理员的事情。

2.2.1 系统的登录

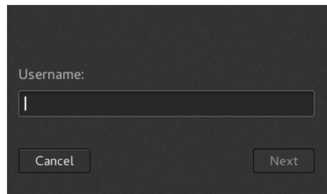
1．图形界面登录

用户可以在如图 2-1 所示的欢迎界面中选择用户名登录。若选择了用户，则在如图 2-5（a）所示界面的 Password 框中再输入正确密码并回车就可以进入系统工作。

在红帽系统中，用户还可以单击“Not listed”来选择未列出的用户登录。方法是单击“Not listed”，然后在弹出的如图 2-5（b）所示的用户名（Username）输入框中输入用户名并回车，再在如图 2-5（a）所示的 Password 输入框中输入用户的正确密码才能登录系统。若在图形界面登录 root 用户，则可以采用这种办法，但是 Ubuntu 的 Unity 环境没有提供图形界面登录 root 的办法。



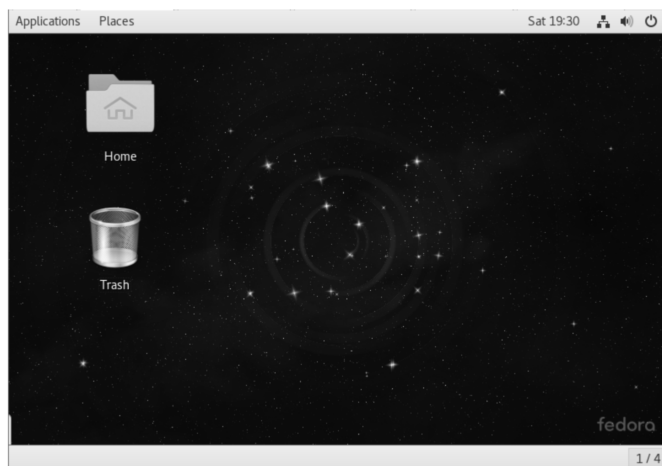
（a）选择用户名后的密码输入界面



（b）选择“Not listed”后的用户登录界面

图 2-5　用户登录界面

在图形界面下，用户成功登录后的桌面系统如图 2-6 所示。



（a）红帽系统的 GNOME 桌面



（b）Ubuntu 的 Unity 桌面

图 2-6　Linux 系统的桌面

若系统安装有多个桌面环境，如红帽系统默认安装有 GNOME、GNOME Classic 和 GNOME on Wayland 等，用户在输入完密码后回车或确认前，可以选择桌面环境。本书关于红帽的描述中使用的都是 GNOME Classic 环境。

若为首次登录，则会出现选择语言、选择键盘、隐私和在线账号设置界面，这些设置完成后进入桌面系统。

用户在图形界面登录成功之后就可以像在 Windows 一样，通过鼠标进行各种操作。

2．字符界面登录

1）登录

字符界面也叫终端界面。字符界面的登录过程与图形界面基本相同，只是所使用的界面不同而已。用户需要在如图 2-2 所示界面中的“login：”后输入用户名，回车后在“Password：”后输入用户密码，最后回车就可以了。以 root 用户为例的登录过程如图 2-7 所示。

这里需要提醒的是，当用户在“Password：”后输入密码时屏幕并不回显，用户只需要在

输入密码完毕后回车就可以了。

```
Fedora 24 (Workstation Edition)
Kernel 4.5.5-300.fc24.x86_64 on an x86_64 (tty3)

localhost login: root
Password:
Last login: Thu Apr 20 10:46:01 on tty2
[root@localhost ~]# _
```

图 2-7　字符界面登录过程

当用户成功登录之后，系统就会根据用户身份的不同而显示相应的提示符，用户可以执行与自己身份和权限相称的命令。

2）提示符

在 UNIX/Linux 系统中，默认情况下，root 的提示符为“#”，一般用户的提示符为“$”。之所以说是默认情况下，是因为系统的提示符是可以改变的。

3）说明

需要说明的是，在 Ubuntu 系统中，root 用户没有被启用，因而不能以 root 身份进行登录。若要启用 root 用户，需要以某具有管理员权限的用户执行命令

```
$ sudo passwd root
```

为 root 用户设置密码后，才能从字符界面登录。为 root 用户设置密码的过程如下：

```
[sudo] passwd for hncj:                    #为 root 设置密码，两遍要一致
Enter new UNIX password:                   #在冒号后输入新密码
Retype new UNIX password:                  #在冒号后再输入一遍新密码
passwd: password updated successfully
```

这样处理之后，在 Unity 界面下，root 用户仍然不能登录（但可在字符界面登录）。既然系统不允许，建议也不要破戒，若需要执行具有管理功能的命令或操作，可使用 sudo 命令（参见第 4 章），这是 Ubuntu 系统的一贯做法。

2.2.2　注销

1．图形界面

单击（如图 2-6 所示）桌面右上角，即顶部面板最右侧的或按钮，会出现如图 2-8 所示的界面。在此菜单界面中选择“注销（Log Out）”，会弹出如图 2-9 所示的“注销”对话框，选择“注销”后即可从系统中退出返回登录界面。对于红帽的 GNOME 界面，在此之前可能还需要单击一下用户名。

2．字符界面

在字符界面下可以通过以下方式注销。

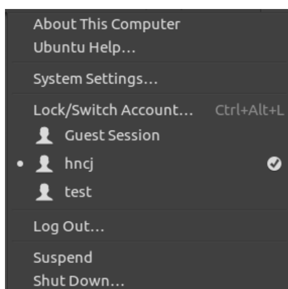
（1）输入 Ctrl+D 组合键。

（2）执行命令 exit。

（3）执行命令 log out。

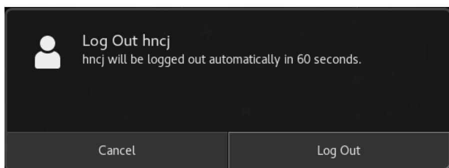


（a）红帽系统 GNOME 状态栏菜单

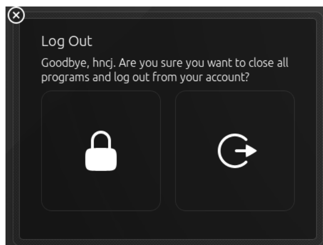


（b）Ubuntu Unity 状态栏菜单

图 2-8　系统状态栏下拉菜单



（a）红帽 GNOME 注销界面



（b）Ubuntu Unity 注销界面

图 2-9　注销界面

2.2.3　CLI 终端仿真界面

在红帽系统的 GUI 下，用户可以通过“Applications（应用程序）”→“System Tools（系统工具）”→“Terminal（终端）”打开一个如图 2-10 所示的终端仿真界面，在 Ubuntu 的 Unity 界面下，也可以快捷键“Ctrl+Alt+T”打开它。

终端仿真界面是在 GUI 下的一个 TUI 模拟界面，不仅可以运行在 TUI 下可以执行的所有命令，而且可以启动能在 GUI 下执行但不能在 TUI 下执行的 X 界面应用程序。该程序的界面是基于命令行的，为了区别于 TUI 和 GUI，人们称它为 CUI 或 CLI，意为命令行界面。

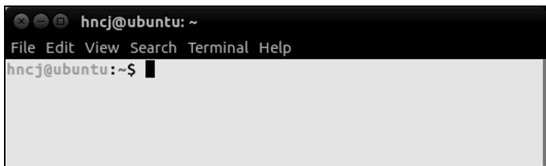


图 2-10　GUI 下的终端仿真界面 CLI

2.2.4　程序的启动

1．从应用程序菜单或快速启动栏启动

若要启动一个任务，红帽系统可从应用程序下拉菜单中选取，Ubuntu 系统可以从快速启

动栏中选取。这种方法直观但不高效，当应用程序较多时，找到程序不易，选准也不易。还有一个问题就是任务或快速启动栏内没有包括所有可用程序。

2．从 TUI 或 CLI 启动

用户可以直接在 TUI 或 CLI 的命令提示符后输入要执行的命令，然后回车就可执行命令。这是专业人员的常用办法。

3．以 Alt+F2 组合键启动

在图形界面按下 Alt+F2 组合键，在弹出的输入框中输入命令及所需要的参数，然后回车，就可启动程序。

2.3　Linux 系统的关闭与重启动

因为 UNIX/Linux 采用了异步写和延迟写机制，大部分数据操作都在缓冲区内进行，并没有实时写入设备，如果强行关机会导致数据丢失，严重者可造成系统瘫痪。因此在切断电源之前必须先关闭系统，决不可断电了事。Linux 系统关闭可以在字符或图形界面下进行。

2.3.1　字符界面

在字符界面下常用的关机/重启命令有 halt、reboot、poweroff、shutdown 和 init 等，它们都可以达到关闭或重启系统的目的，但每个命令的工作过程是不同的。

1．halt 命令

halt 命令的功能是关闭系统，执行过程中，终止所有应用和系统进程，将所有数据写入存储介质，最后关闭系统。其常用方法为：

```
halt [-f] [-p]
```

参数-f 用于强制快速关机或重启；参数-p 用于在关闭系统时关掉电源。

一般情况下，使用-p 参数时会在关闭系统后自动关闭电源。如果计算机不这样做，在看到消息“System halted.”后，可以手动关掉电源。

2．reboot 命令

reboot 的工作过程与 halt 几乎一样，但其在关闭系统时会重新启动它。

3．poweroff 命令

poweroff 命令等价于 halt -p，关闭系统时同时关掉电源。

4．shutdown 命令

前几个命令用于关机时太干脆了，也太武断了，命令一经发出就开始执行且不能反悔。shutdown 命令可以安全地关闭系统，并在真正执行系统关闭与命令发出之间可以指定一个时

间延迟，以供用户做好准备并从容退出。使用 shutdown 命令时，将在关机前向系统内的所有用户发送通知或警告信息，告诉系统发生了什么及如何处理，如系统要在什么时候关机或重启，请做好准备，并提前退出等。其用法为：

```
shutdown [-krhfFc] time [wall msg]
```

shutdown 命令的部分参数如表 2-1 所示。

表 2-1　shutdown 命令的部分参数

选　　项	功 能 描 述
-r	重启计算机
-h/-H	关机后关闭电源
-k	并不真正关机，只是发送警告信号给每位用户
-c	取消目前已经提交的 shutdown 任务。该选项同时使用可有一个用来解释撤销关机的广播信息
time	指定关机时间。可为格式为 hh:mm 的绝对时间，如 1:20，或相对时间（单位为分），如+10 表示 10 分钟后关机，+0 表示 now。若不指定则默认为+1
wall msg	广播信息。若不指定，系统提供默认信息

说明：当不带任何选项，比如，-h、-H、-r 执行 shutdown，默认切换到运行级 1，即单用户模式；若要通过 shutdown -c 撤销已经发出的 shutdown 任务，则在以前的系统中，必须在与发出 shutdown 命令不同的终端上进行，现在不必了。

当指定时间在 5 分钟以内，或已经到达距关机不足 5 分钟的时候，系统就会产生 /run/nologin 文件，拒绝用户登录。

shutdown 的应用示例如下：

```
#shutdown -r now                                      #立刻重新启动
#shutdown -h now                                      #立刻关机
#shutdown -k now 'Hey Lets go now.'                   #发出警告信息，但没有真关机
#shutdown -h 10:42 "系统将在 10:42 关闭，请届时退出"  #10:42 分关机
#shutdown -r +20 '20 分钟后将重启系统，请提前退出'    #20 分钟后重启系统
#shutdown -c                                          #撤销已下达的 shutdown 命令
#shutdown now                                  #切换至单用户模式（不加任何选项时）
```

2.3.2　系统的运行级别及切换

1．系统的运行级别

Linux 系统有 7 个运行级或运行状态，定义如下。

0：关闭系统。

1：单用户模式。

2：多用户模式，但没有 NFS 功能。

3：完全多用户模式。

4：没有使用，用户可自定义。

5：完全多用户模式，且支持 X-Windows（系统默认运行级）。

6：重新启动。

系统的运行级可通过命令 man runlevel 做进一步了解。

上述的 0 和 6 不能作为真正的运行级别，因为关机和重启只能作为状态来存在。系统默认的运行级通常为 5。系统正在运行的级别可以通过以下命令来查询：

```
#who -r                    #或
#runlevel
```

2．运行级别的切换（init 和 telinit）

运行级别的切换是通过 init 和 telinit 命令来实现的。系统内运行着一个叫 init 的进程，此 init 在引入 systemd 软件包之后叫 systemd，它负责系统的初始化和运行级别的切换。命令 init 或 telinit 的作用是告诉 init 进程进行运行级切换，在使用时，init 和 telinit 可视为同一个命令，其用法为：

```
init LEVEL                 #LEVEL 为数字，7 个运行级别中的一个
telinit LEVEL
```

init 命令的使用示例如下：

```
#init 0                    #关机
#init 6                    #重新启动
#telinit 1                 #切换到单用户
```

2.3.3　图形界面

在图形界面下，用户可以通过 2.2.2 节注销的办法打开如图 2-8 所示的界面，然后单击“关机按钮 ”或“Shut Down”，最后在弹出的如图 2-4 所示的界面中选择“Power Off（关机）”或“Restart（重启）”进行关机、重启系统，当然也可以选择“Cancel（取消）”取消操作。

2.4　Linux 系统的图形界面介绍

图形界面（X-Window）就是在 Linux 操作系统中提供图形化的用户界面，支持的视窗系统也被称为 X。X-Window 的工作方式跟 Microsoft Windows 有着本质不同。MS（Microsoft）Windows 的图形用户界面与操作系统本身紧密结合，成为操作系统的一部分；而 X-Window 并不是操作系统的一部分，它实际上只是在 Linux 操作系统上面运行的一个应用程序，可以不启动。换句话说，MS Windows 的图形支持是内核级的，而 Linux 的 X-Window 是应用程序级的。

X-Window 的一个主要特性就是它采用了“客户端-服务器”模式。其组成由 X 服务器（X Server）、X 客户端（X Client）和通信通道（comunication channel）三部分组成，XServer 和 XClient 可位于同一台主机上，也可独立地运行于同网络上的不同主机。

X-Window 只是一个框架，本身只定义了最基本的窗口功能，如建立窗口、鼠标控制和键盘输入等，实际上所有显示在 X 屏幕上的外观及控制等界面都要由窗口管理器来管理。

所谓桌面环境就是包括窗口管理器、面板、桌面，以及一整套应用程序和系统工具在内的套件。Linux 环境下广泛使用的桌面环境是 GNOME（GNU Network Object Model

Environment）和 KDE（K Desktop Environment）等。红帽系列系统默认的桌面系统是 GNOME，Ubuntu 默认使用的是 Unity 桌面环境。

2.4.1　GNOME 桌面简介

GNOME 默认包括两个面板和一个桌面，用户成功登录后的 GNOME 图形桌面环境如图 2-6 所示。

1．GNOME 的面板

默认情况下，经典 GNOME 桌面的两个面板分别位于屏幕顶部和底部，这样的设计可以让用户方便地启动应用程序、管理正在运行的程序和在多个虚拟桌面间切换。默认情况下，顶部面板（Top Panel）包含了如图 2-11 所示的两个菜单按钮、活动窗口、小程序和状态区，两个菜单分别是“应用程序（Applications）”和“位置（Places）”；底部面板（Bottom Panel）包含了如图 2-12 所示的一个活动程序托盘和工作区切换器。

用户也以创建新的面板或删除已经存在的面板。

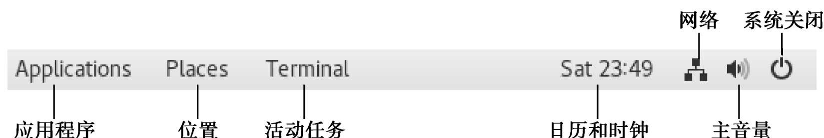


图 2-11　顶部面板



图 2-12　底部面板

2．顶部面板

顶部面板由菜单区、活动窗口区、小程序和状态区构成。

1）两个菜单

顶部面板包括两个菜单按钮，它们都是下拉菜单，其中所包含的功能项目与所安装的软件包或应用程序的多少有关，用户可以从中选择相关程序或功能进行操作。“应用程序（Applications）”菜单包含的是当前用户可以使用的应用程序和系统工具等，其中系统工具的大多数与系统管理相关的操作都需要有超级用户权限方能实施；“位置（Places）”菜单中包含的是与当前用户有关的资源信息，如用户主目录内的常用目录、桌面上的资源、网络共享资源和文件资源查找等。

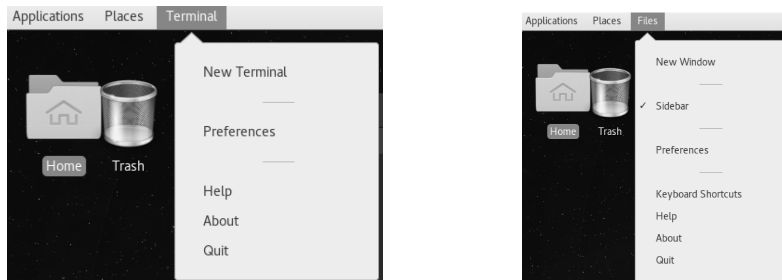
2）活动窗口

活动窗口仅显示当前处于活动状态的任务窗口菜单（见图 2-13），通过此菜单可以设置或查看该任务窗口的相关信息。

3）小程序和状态区

顶部面板的最右端为小应用程序和系统状态区，用户可以单击其中的项目打开相关项目进行状态观察或进行某种设置。系统状态区包含的内容多少与系统的硬件和所安装的软件均

有关，如笔记本系统中会有电池状态等。通过小程序和状态区，可以查看和设置日历和时钟、主音量、网络（有线、无线和蓝牙等）、锁屏和关机，还可以打开系统设置中心进行系统设置和管理。



（a）活动窗口为终端的菜单界面　　（b）活动窗口为主目录的菜单界面

图 2-13　活动状态的任务窗口菜单

3．底部面板

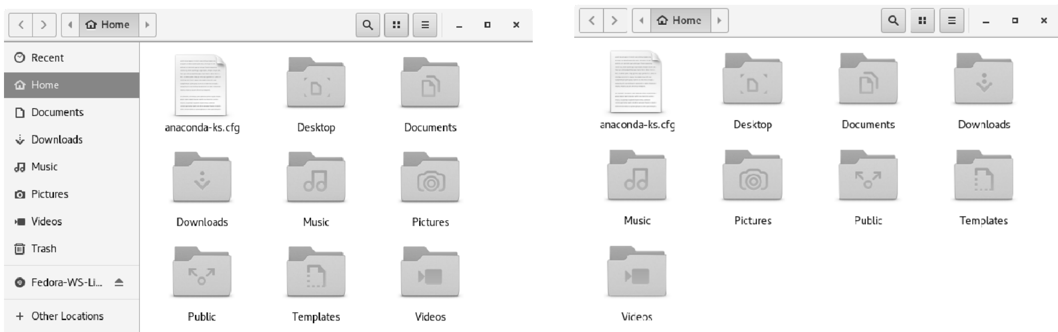
底部面板较顶部面板简单一些（见图 2-12），其中只包含“活动任务区”和“工作区切换”等。

（1）活动任务区。活动任务区排列着操作者前台正在运行的程序图标，通过单击这些图标可以极小或极大化任务窗口。

（2）工作区切换。默认情况下，当用户从图形界面登录成功后，系统为用户设置 4 个工作区。默认时用户工作在第 1 个工作区，用户可以单击1/4按钮打开工作区切换菜单点选其他工作区实现工作区的切换。在每个工作区内，用户都可运行相同或不同的应用程序。

4．桌面

系统的桌面如图 2-6 所示，上面可以排列各种图标和应用程序窗口，应用程序窗口可被极小化到活动任务区。打开的“Home（主目录）”窗口如图 2-14 所示。



（a）带侧边栏的主目录窗口　　（b）不带侧边栏的主目录窗口

图 2-14　打开的主目录窗口

5．Nautilus 文件浏览器

在 Linux 系统中，每当打开一个资源窗口时，所使用的应用程序都是 Nautilus。Nautilus

的使用类似 Windows 的资源管理器，它按文件夹组织文件和进行如下操作。

（1）创建和显示文件夹和文档：创建新文件、按文件夹组织文件和保存文件。

（2）搜索和管理文件：为文件分级并按其分级搜索。

（3）打开计算机的特殊位置：访问本地网络并保存文件。

（4）向 CD 或 DVD 写入数据：计算机包含全部驱动器和文件系统，使得将文档备份到 CD 和 DVD 变得特别容易。

（5）使用浏览模式导航：以此模式打开文件夹，文件夹将会在同一窗口内打开。

Nautilus 的资源窗口由“位置栏”、“工具栏”、“主窗体”和“侧边栏”组成。

“位置栏”位于界面顶端的左端，以层级文件夹方式显示已打开文件夹的当前位置，在浏览模式中，可以转到包含当前文件夹的父文件夹。要转到父文件夹，可单击位置栏中层级文件夹中的父文件夹，也可单击导航工具栏上的<或>在父子文件夹间切换。

“侧边栏（Sidebar）”显示存储于计算机的其他文件夹。“侧边栏”是不常驻的，可以在活动窗口区的菜单“Files（文件）”（见图 2-13（b））中单击“Sidebar（侧边栏）”，则可以隐藏或显示侧边栏（见图 2-14）。默认情况下 Nautilus 窗口带有侧边栏。

“工具栏”位于界面顶端的右方。每个打开的 Nautilus 窗口都有按钮、和，用于文件搜索、排序和文件夹处理相关操作。

“主窗体”位于界面的右下方，是拥有空间最大的一个，其中存放着所选位置的内容（文件夹或文件）。对主窗体内文件或目录的操作请参照 MS Windows 系统。

用户还可以对 Nautilus 进行个性化设置。在菜单“Files（文件）”中单击“Preferences（首选或偏好）”，则可以打开首选项设置框（见图 2-15），用于对视图（Views）、行为（Behavior）、列表列（List Columns）、搜索和预览（Search & Preview）的处理。在菜单“Files（文件）”中单击“Keyboard Shortcuts（键盘快捷键）”，则可以打开键盘快捷键对话框，用于查阅与 Nautilus 文件管理器操作有关的快捷键（图略）。

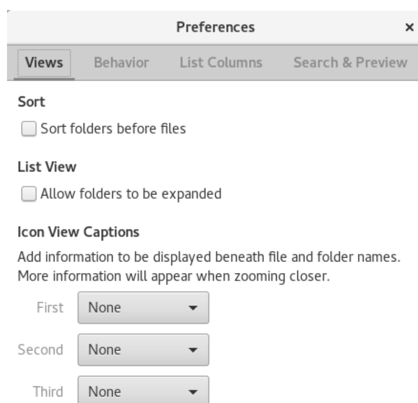


图 2-15　Nautilus 首选项设置框

2.4.2　Ubuntu 的 Unity 桌面简介

Ubuntu 默认使用的是 Unity 桌面环境，当用户登录进入之后，可以看到 Unity 桌面（见图 2-6（b））。默认情况下，它由顶部面板、左边快速启动栏和位于顶部面板之下、快速启动

栏之右的主桌面组成。

顶部面板与 GNOME 的类似，右端是小程序和状态区，左端为活动窗口区。

左侧的快速启动栏中只包含了很少一部分启动图标或活动任务，单击其中的图标可启动一个任务，或让一个正在执行的任务获得焦点而变为当前任务。用户可从其上删除图标，或者向其添加启动图标。若要从其中删除一个项目，可右击欲删除的项目，并在弹出的菜单中选择“Unlock from Launcher”就行。若向其上添加启动项，最简单的方法是，首先启动一个任务，则该任务图标将出现在快速启动栏中，然后单击该图标，在弹出的菜单中选择“Lock to Launcher”就可以了。

Unity 桌面上的文件管理与操作与 GNOME 下相同，因为 Ubuntu 使用的文件管理工具也是 Nautilus。

Unity 只是 Ubuntu 的默认界面，在 Ubuntu 下也可安装 GNOME 界面。若在 Ubuntu 下安装了 GNOME 界面，则本书中介绍的几个系统的操作界面就相同或接近一致了。

2.4.3　设置中心（Settings）

红帽系统下，通过菜单“Applications（应用程序）”→“System Tools（系统工具）”→“Settings（设置）”，或者单击顶端面板最右侧的并从其下拉菜单中选择，或者在 CLI 界面下执行 gnome-control-center 命令，都可以打开系统“设置中心”（见图 2-16）。在 Ubuntu 下的启动方法是从快速启动栏单击，或单击顶端面板最右侧的从其下拉菜单中选择“System Settings”，或者在 CLI 界面下执行 unity-control-center 命令。设置中心界面分为三栏：Personal（个人）、Hardware（硬件）和 System（系统）。有管理权限的用户可以选择相关项目进行设置和管理。

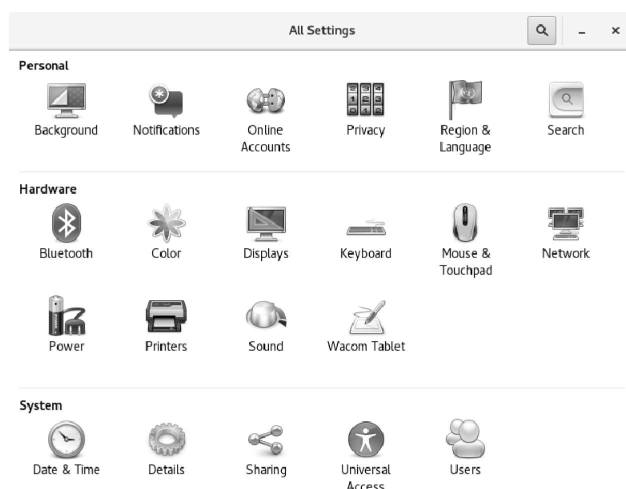


图 2-16　系统设置中心

1．设置系统偏好或首选项

Linux 系统允许用户改变桌面及系统的偏好或首选设置，设置方法为：在打开的系统设置中心个人栏中，选择具体的功能项目设置。比如，用户可以在“Background（背景）”功能

中选择“Background”或者“Lock Screen”进行桌面背景或锁屏背景的设置；还可以在“Privacy（隐私）”功能中设置“Screen Lock（锁屏）”、“Location Services（位置服务）”、“Usage & History（用量及历史）”、“Purge Trash & Temporary Files（清理回收站及临时文件）”，以及“Problem Reporting（报告问题）”的开启与关闭等，在“锁屏”中开启“Automatic Screen Lock（自动锁屏）”时还可以设置“Lock screen after blank for（黑屏至锁屏的等待时间）”等。

设置偏好或首选可以带来特殊效果，但也会带来不必要的麻烦，并且需要额外的知识，用户应根据需要和能力进行设置，这里不再介绍。

2．设置应用程序快捷键

使用快捷键可以快速启动应用程序，在系统设置中心硬件栏“Keyboard（键盘）”功能项目中，系统预先定义了一些应用程序的快捷键，用户可以使用这些快捷键启动应用程序，也可以禁用或者修改这些应用程序的快捷键；用户还可以添加其他应用程序的快捷键来启动应用程序。不建议修改系统已经设置好的“默认”或“通用”快捷键。

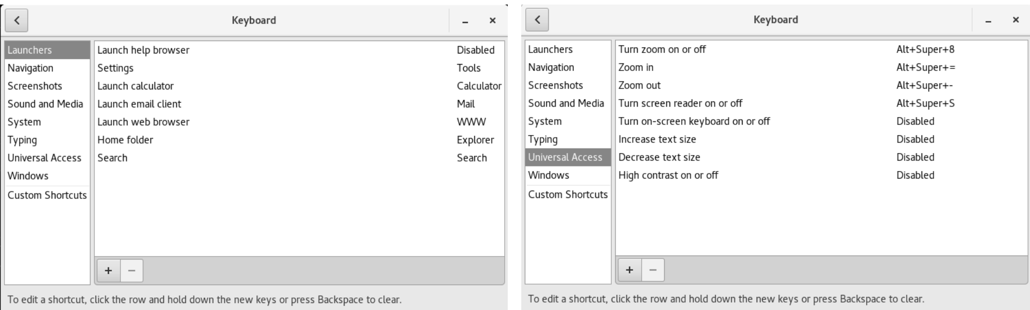
根据应用程序的类型，系统预定义快捷键的应用程序有 8 类：Launchers（启动器）、Navigation（导航）、Screenshots（截图）、Sound and Media（声音和媒体）、System（系统）、Typing（打字）、Universal Access（通用辅助功能）及 Windows（窗口），如图 2-17（a）所示。如在“启动器”类型中，系统预定义快捷键的应用程序有 Launch help browser（启动帮助浏览器）、Settings（设置）、Launch calculator（启动计算器）、Launch email client（启动邮件客户端）、Launch web browser（启动网页浏览器）、Home folder（主文件夹）和 Search（搜索），用户可以对这 7 个应用程序进行快捷键禁用或重设。

1）快捷键设置示例

下面以“通用辅助功能”类型中应用程序“Turn on-screen keyboard on or off（开关屏幕键盘）”的快捷键修改进行介绍。

（1）在打开的系统设置中心硬件栏中，选择“键盘”功能项目，打开键盘对话框（见图 2-17（a））。

（2）选择左侧应用程序类型栏中的“通用辅助功能”，右侧应用程序栏中有 8 个预定义快捷键的应用程序，如图 2-17（b）所示。其中“开关屏幕键盘”的快捷键目前为“Disabled（禁用）”。



（a）选择启动器的键盘对话框　　（b）选择通用辅助功能的键盘对话框

图 2-17　键盘对话框

（3）单击该应用程序，则“Disabled（禁用）”显示为“New accelerator…（新建加速键…）”，

此时，输入按键组合即可设定其快捷键，如输入“Alt+Super+k”（注：Super 键即键盘上的 Win 键）。输入退格键（Backspace）则快捷键设为禁用；输入取消键（Esc）则取消本次设定。

快捷键设定为“Alt+Super+k”后，选择“应用程序”→“附件”→“文本编辑器”，打开文本编辑器，即可使用快捷键“Alt+Super+k”打开或关闭屏幕键盘（图略）。

2）添加快捷键示例

用户除了可以对预定义快捷键的应用程序进行快捷键重设外，还可以对其他应用程序设定快捷键。对其他应用程序设定快捷键可以使用键盘对话框应用程序栏下方的按钮 + 进行添加。下面来增加一个新应用程序“终端”的快捷键。

（1）在打开的系统设置中心硬件栏中，选择“键盘”项目，打开键盘对话框（见图 2-17（a））。

（2）单击 + 按钮，打开自定义快捷键对话框，如图 2-18（a）所示。在“Name（名称）”文本框中输入自己给应用程序“终端”所起的名字，如 gterm；在“Command（命令）”文本框中输入“终端”应用程序的启动命令“/usr/bin/gnome-terminal”，如图 2-18（b）所示。



（a）输入命令前自定义快捷键对话框　　（b）输入命令后自定义快捷键对话框

图 2-18　自定义快捷键对话框

（3）单击“Add（添加）”按钮，则在“Custom Shortcuts（自定义快捷键）”类型的应用程序栏中新增一项名称为“gterm”的应用程序（图略），其快捷键为“Disabled（禁用）”。

（4）按照修改预定义快捷键应用程序的方法，为新增的应用程序“gterm”设定其快捷键，如“Ctrl+Alt+T”。

快捷键设定完成后，即可使用快捷键“Ctrl+Alt+T”打开“终端”应用程序（见图 2-18）。

说明：在 Ubuntu 系统的 Unity 环境中，对 CLI 界面的此快捷键设置并不必要，因为在 Ubuntu 系统安装时已经预设好了。但是，对于 Ubuntu 系统来讲，快捷键设置似乎更有意义，因为 Unity 的快速启动栏空间有限。

2.5　Linux 系统的在线帮助与资源

UNIX/Linux 系统比较规范，各种软件都有完整的开发和使用说明文档，用户可以通过多种渠道获得系统的帮助。

2.5.1　man

1．功能及用法

man 是传统 UNIX 系统的在线手册页，通过它用户可以获得在线帮助。man 是 manual 的缩写，使用 man 命令可以获得相关主题的帮助。man 的常用方法为：

```
man [-afhw] [sec] name …
```

2．参数说明

man 命令的常用参数如表 2-2 所示。

3．手册页的组织

Linux 系统的手册页的组织如表 2-3 所示，用户可根据需要在使用 man 命令时指定章节。

表 2-2　man 命令的常用参数

参　数	功 能 描 述	参　数	功 能 描 述
sec	指定章节	name	指定帮助的主题
-a	对所有与 name 相关的内容进行帮助	-f	列与 name 相关的描述信息，同 whatis
-h	显示帮助信息	-w	显示手册页的位置

表 2-3　手册页的组织

章　节	内　　容	章　节	内　　容
1	普通用户指令部分	6	游戏
2	系统调用部分	7	协议转换和杂项
3	C 库函数	8	系统维护命令（只有超级用户可以使用）
4	设备及特别文件	9	内核
5	文件格式及轮换	n	新增部分

man 手册页文件存放在/usr/share/man/man?目录下，?是章节数，文件名命名规则为：

```
手册名称.章节号.gz
```

可用 man -w name 来查询文档位置，使用 whatis name 或 man -f name 查询 name 的功能及描述信息。

4．使用举例

```
#man -w passwd          #查询首个与 passwd 相关的帮助文件
#man -aw passwd         #列出所有与 passwd 相关的帮助文件
#whatis passwd          #查询对 passwd 的简单说明
#man -f passwd          #同上
#man 1 passwd           #命令用法
#man 5 passwd           #文件格式
#man -a passwd          #全部
```

当用户使用 man 命令阅读某个项目的手册页时，系统进入 man 操作界面。可以使用的部分功能键如表 2-4 所示，也可以使用光标键移动位置。

表 2-4　man 的功能键

键	功 能 描 述	键	功 能 描 述
q	退出	h	显示帮助

（续表）

键	功 能 描 述	键	功 能 描 述
Home	首页	End	尾页
Space /PgDn/D/^D	下翻一页	PgUp/b/^B	上翻一页
/string，?string	查找字符串	n	继续查找
Enter	下翻一行	：!/!<cmd>	执行 shell 或命令 cmd

man 工作时需要几个程序的配合，其中的分页命令在早期的 UNIX 系统中为 more，在现在的 Linux 系统中为 less，表 2-4 中的功能键就是分页命令的部分功能键。

2.5.2 textinfo

textinfo 是 Linux 系统提供的另外一种格式的帮助信息。和 man 相比，textinfo 具有更好的交互功能。它支持链接跳转功能。通常使用 info 和 pinfo 命令来阅读 textinfo 文档。

运行 info，可以在 shell 提示符后输入 info（不要参数），它将列出一个文档的清单。如果没有发现你所需要的，那可能是因为没有安装包含那个文档的软件包。info 帮助系统的初始屏幕显示了一个主题目录，用户可以将光标移动到带有“*”的主题菜单上，然后按回车键进入该主题，也可以输入 m，后跟主题菜单的名称而进入该主题。例如，输入 m，然后再输入 gcc 就会进入 gcc 主题中，也可指定帮助项目，如 info gcc。

info 系统是一个超文本系统。任何高亮度显示的文字都有一个链接导向信息。使用“Tab”将光标移到链接，并按“Enter”进入链接。按“p”返回上一页，按“n”翻到下一页，按“u”回到文档的上一层，按“q”退出 info。

pinfo 兼容 info 的功能，并且支持彩色链接文本、鼠标选定支持等功能。其用法与 info 基本相同。pinfo 提供的基于浏览器风格的 textinfo 的文档程序界面，使得操作更加简单，界面更加友好。

2.5.3 yelp

在 GNOME 桌面环境下提供有专门的帮助功能 yelp，通过 yelp 不仅可以在图形环境查看手册页和 textinfo 帮助文档，还可以浏览 GNOME 桌面本身的联机帮助文档。此外 yelp 还提供了索引功能，可以在各种文档中进行跳转查找，提高用户获得帮助信息的速度。运行 yelp 的方法是：

（1）在 GNOME 的“系统”菜单上选择“帮助”。

（2）在 CLI 界面执行命令 yelp。

2.5.4 Linux 系统的其他帮助和资源

1．软件包项目文档

Linux 中的大多数软件开发工具都是来自 GNU 项目，这些软件包除了提供手册页和 textinfo 外，还提供项目文档。每个软件包都拥有与其同名的文档目录，目录命名格式为：“软

件包名称-版本号”。有些项目文档为了方便阅读还提供了多种格式的文件。html 格式用浏览器阅读，ps 格式用 postscipt viewer 查看或打印输出，txt 格式用文本编辑器查看或在命令行下阅读。软件包的项目文档都包含特定的文件，文件命名遵循规定：AUTHOR：作者信息；BUGS：系统存在的 Bug；CHANGES：修改记录；COPYING：版权声明；INSTALL：安装方法；LANGUAGES：语言类型；NEWS：新版本的特点；ONLINEHELP：在线帮助；TODO：项目发展计划。

2．HOWTO 文件

HOWTO 文件是可供用户参考的联机文档的另一种形式，HOWTO 文件的文件名都有一个 HOWTO 后缀，并且都是文本文件。每一个 HOWTO 文件包含 Linux 某一方面的信息，例如，它支持的硬件或如何建立一个引导盘等。

3．LDP 文档

从第一个 Linux HOWTO——“安装 HOWTO”开始，Matt Welsh 首创了面向解决方案、要点提纲形式的文档。如今，这已经成为像 Linux 一样的工程项目，越来越多的人加入其中，相应的工具出现了，文档也大量涌现，并被翻译成各国语言发布。LDP 是最大的 Internet 工程之一，它拥有数百成员撰写的数百个文档，包括小到简明的手册页和大到上百页的详细指南。这些文档几乎覆盖了 Linux 的所有方面，并且免费发布。

LDP 的主页是 http://www.tldp.org/。提供的文档类型如下。

（1）HOWTO 文档：HOWTO 文档采用多种格式进行发行，如 txt、html、pdf 等流行的文档格式都有，读者可以根据需要下载相应的文档压缩包进行离线阅读，HOWTO 的作者会根据需要不定期地对文档进行版本更新，通常文档的更新是跟随某个相关软件或发行版本的升级而进行的。

（2）FAQ 文档：FAQ 文档采用一问一答的形式对常见的问题给出解决方法，往往能够在找到问题之后快速解决问题。

（3）Guide 文档：Guide 文档是对 Linux 某个方面的应用指南，主体范围较大、篇幅较长，读者可以把它作为比较系统的教程来学习。

2.6　LibreOffice 办公套件简介

Linux 的发行套件包括一个功能强大的办公套件 LibreOffice。LibreOffice 是 OpenOffice.org 的现代衍生产品，它把几个互补的应用程序综合成一个软件包。LibreOffice 允许用户完全控制文档的内容和布局，还可以让用户边编辑边查看效果（所见即所得）。

LibreOffice 套件包含几个用于文字处理、电子表格、演示文稿和绘图的软件。表 2-5 显示了可以在 LibreOffice 套件中使用的多种不同文件类型及可以用它来完成的任务。

表 2-5　LibreOffice 的功能与特性

应 用 程 序	文件兼容性	文 档 类 型
Writer	.sxw, .sdw, .doc, .rtf, .txt, .htm/.html	正式公函、商业表格、学术论文、简历、新闻简报、报告

（续表）

应 用 程 序	文件兼容性	文 档 类 型
Calc	.sxc, .dbf, .xls, .sdc, .slk, .csv, .htm/.html	电子表格、图表、表格、人事通讯录、地址簿、收据和账单、预算、简单数据库
Impress	.sxi, .ppt, .sxd, .sdd	商业和学术演示文稿、万维网演示、演讲、幻灯片放映
Draw	.sxd, .sda；可以被导出为多种格式，包括:.jpg, .bmp, .gif，和 .png	图示、线条绘图、剪贴图片、机构图表

LibreOffice 套件具有许多文件兼容性功能，并能够允许用户为学术、商业或家庭使用完成相关任务。由于 LibreOffice 套件与微软 Office 系列的相关软件在功能和使用上有许多相似之处，这里只对 LibreOffice 套件做简单介绍。

2.6.1 LibreOffice Writer

LibreOffice Writer 是一个功能强大的文字处理器，相当于 Windows 的 Word。可用以下方式启动 LibreOffice Writer。

（1）按顺序选择“Applications（应用程序）”→“Office（办公）”→“LibreOffice Writer”。

（2）在图形界面下双击一下具有合适扩展名的文档，如.doc。

Writer 主界面是文档编辑区域，用户可以在其中输入文本。在输入过程中可以使用“Ctrl+Space”进行中英文切换用，使用“Ctrl+Shift”进行输入法切换用。在窗口顶部的工具栏里聚集了各类功能工具，它们可以控制字体、纸张大小、版面调整（左、右或居中对齐）等。其中，还有一个文本箱，用户可以在里面输入文档的确切位置来将它载入编辑区域。工具栏上还有打开、保存和打印文档的按钮，以及创建新文档的按钮。

窗口中工具栏上的按钮可以用来检查拼写、自动突出显示拼错的单词、单词或词组搜索，以及其他便利的编辑功能。如果鼠标在某一工具栏按钮上徘徊，一个弹出的提示信息会显示该按钮功能。用户可以单击“帮助”菜单，选择“说明文”来进行详细帮助。

用户可以使用默认设置在文档编辑区域输入文本，并可打开或另存为 Word 文档，还可以把文档导出为 HTML 或 PDF 格式。

Writer 不但可用于常规的文本编辑，还可以在文档中添加图像、图示、图表及表格对等，使其图文并茂。

2.6.2 LibreOffice Calc

LibreOffice Calc 是一个软件电子表格应用程序，类似于 Windows 的 Excel，它允许用户在组织成行列的单元格中输入和处理数据。单元格是单个数据的容器，如数量、标签或数学公式。用户可以执行一组单元格的计算，或根据包含在一组单元格中的数值来创建图表。用户甚至可以把电子表格的数据融入文档来增加专业化色彩。可按以下方式启动 LibreOffice Calc。

（1）按顺序选择“Applications（应用程序）”→“Office（办公）”→“LibreOffice Calc”。

（2）在图形界面下双击一下具有合适扩展名的文档，如.xls。

Calc 允许用户输入并处理个人或商业数据，并可通过函数完成计算功能等。如果用户需

要创建图表或图例，可以选择使用图表模板。

用户可以把用 Calc 创建的电子表格保存为好几种文件格式，包括原始的.sxc 和与 Microsoft Office 兼容的.xls 格式。此外，也可以把绘制的图标和图例导出到好几种图像文件格式，并把它们综合到文档文件、网页和演示文稿中。

2.6.3 LibreOffice Impress

LibreOffice Impress 是一个能够帮助用户制作演示文稿的图形化工具，相当于 Windows 的 PowerPoint。可使用以下方法启动它。

（1）按顺序选择“应用程序”→“办公”→“LibreOffice Impress”。

（2）在图形界面下双击一下具有合适扩展名的文档，如.ppt。

Impress 中有一个自动文件辅助的渐进演示向导，允许用户从样式模板集合中创建演示文档。用户可以创建带有逐项列表、轮廓或图像的幻灯片，甚至可以把 LibreOffice Calc 中的图表导入幻灯片。

当用户首次启动 LibreOffice Impress 时，会看到自动文件助理设置屏幕。用户可以选择幻灯片的风格、演示幻灯片的介质（普通纸、幻灯用透明纸或显示器），以及想在计算机上演示幻灯片时要应用的视觉效果。使用自动文件助理工具选定了首选项后，便可以选择要创建的幻灯片类型创建演示文稿。用户还可以通过选择“演示文稿”→“放映演示文稿”来随时预览演示文稿。

演示文稿可以被保存为几种文件格式，包括原始的 LibreOffice Impress 格式（.sxi）、Microsoft PowerPoint 格式（.ppt）或 StarImpress 格式（.sdd）等。用户还可以按顺序选择“文件”→“打印”把演示文稿打印到普通或透明纸上。

2.6.4 LibreOffice Draw

LibreOffice Draw 能够描画图形，并将它们保存为多种格式。可以按以下方法启动 LibreOffice Draw：

（1）按顺序单击“应用”→“办公”→“LibreOffice Draw”。

（2）在图形界面下双击一下具有合适扩展名的文档，如.sxd。

LibreOffice Draw 有创建直线、曲线、圆、方块之类的基本形状和圆锥、立方体之类的三维物体等的工具栏。用户可以创建图像，并使用主工具栏上的“平面样式/平面填充”下拉菜单中的颜色来填充。Draw 还允许用户打开和导入图像，并使用所提供的工具来修改它们。

当用户完成了对图像或图形的修改后，可以把它保存为多种格式，或导出为.jpg 或.png 之类的通用格式。

习题 2

1．思考题

（1）Linux 系统有哪些运行级别？其含义是什么？

（2）Linux 系统下经常使用的两种桌面环境是什么？

（3）什么是 X-Window 系统？它有什么特点？

（4）Linux 默认提供几个虚拟桌面？如何将一个运行的应用程序从一个虚拟桌面移动到另一个虚拟桌面？

（5）如何进行本地登录和注销？

（6）默认情况下，超级用户和普通用户的登录提示符分别是什么？

（7）如何正确地关闭和重新启动 Linux 系统？

（8）如何获得命令帮助？man 命令的作用是什么？

（9）如何获得 Linux 文档？

2．选择题

（1）字符界面下，使用 init 命令关机所用的参数是（　　），重启系统所用的参数是（　　）。

A．0　　B．1　　C．6　　D．2

（2）字符界面下，使用 shutdown 命令关机所用的参数是（　　），重启系统所用的参数是（　　）。

A．-h　　B．-t　　C．-k　　D．-r

（3）使用 man 命令调阅相关帮助信息时，用于一页页下翻的键是（　　），单行下翻的键是（　　）。

A．Enter　　B．q　　C．b　　D．Space

（4）read 在手册页的多个地方存在，能够查询 read 所有所在章节的命令是（　　），对 read 系统调用进行帮助的命令是（　　）。

A．man -f read　　B．man 2 read　　C．man -a read　　D．man 3 read

实验 2

1．系统的开启与关闭。

2．两种界面的登录、注销与界面切换。

3．系统帮助。

4．图形界面的使用，比较与 Windows 系统的异同。

5．分别访问 Feadora、CentOS、RHEL 和 Ubuntu 网站，查阅相关技术文档。

第 3 章　shell 与 shell 基本操作

3.1　shell 基本功能与基本概念

shell 作为操作系统的外壳，为用户提供使用操作系统的接口。在 UNIX/Linux 中有多种 shell，但在大多数情况下，默认使用的 shell 是 bash 或 bsh（简称 sh）。本书所用系统默认的 shell 均为 bash（Bourne again Shell），它包含了 bsh 的所有特性。

3.1.1　shell 基本功能

shell 是用户和 Linux 内核之间的接口程序，作为人机交互的桥梁，其功能有 6 方面：命令解释执行、文件名及各种替换、I/O 重定向、通信管道建立、系统环境设置和 shell 编程。我们将在本章及以后的章节中逐个展示这些功能。

首先，shell 是一个命令解释器，负责人机交互，接收并解释用户输入的命令，然后将它传送到系统内核，由内核执行完毕后，再将结果按指定格式传递给用户或显示到终端屏幕上。

shell 的另一个重要特性是它自身就是一个解释型的程序设计语言。shell 程序设计语言支持绝大多数在高级语言中能见到的程序元素，如函数、变量、数组和控制结构等。任何在提示符中能输入的命令，都能放到一个可执行的 shell 脚本程序中作为一个语句来使用。shell 编程将在第 10 章专门介绍，shell 的其他功能将在稍后陆续介绍。

3.1.2　字符与保留字

1．字符

shell 中除使用普通键盘可以输入的字符外，还有一些具有特殊含义和功能的特殊字符。使用它们时，应注意其特殊的含义和作用范围。

1）白空格

在 UNIX/Linux 系统中，空格和 Tab 称为白空格（White Space），主要用于命令行中命令名、参数及选项的分隔。在特殊情况下，白空格中也可包含回车字符。

2）通配符

通配符用于模式匹配，如文件名匹配、路径名搜索、字符串查找等。常用的通配符有“*”、“?”和括在方括号“[]”中的字符序列。用户可以在作为命令参数的文件名中包含这些通配符，构成一个所谓的“模式串”，在执行过程中进行模式匹配。

（1）*：代表从它所在位置开始的任何字符串，例如，“f*”匹配以 f 打头的任意文件，但应注意，文件名前的圆点“.”和路径名中的斜线“/”必须显式匹配。例如，“*”不能匹配.file，而“.*”才可以匹配.file。

（2）?：代表它所在位置上的任何单个字符。

（3）[ ]：每次只匹配“[]”中字符的一个。方括号中的字符范围可以由直接给出的字符

组成，也可以由表示限定范围的起始字符、终止字符及中间的连字符“-”组成。例如，f[a-d]与 f[abcd]的作用相同。[ ]内的第一个字符若为“^”或“!”，表示非运算，意为不匹配[ ]内的任何字符，例如，[!a-d]或[^a-d]表示不匹配方括号内 a-d 的字符集，但匹配[ ]之外的字符集。

表 3-1 给出了通配符的具体含义及实例。

表 3-1　通配符含义举例

模　式　串	意　　义
*	当前目录下所有文件名，但不包括“.”开头的文件。例如，abc 是，但.abc 不是
Text	当前目录下所有文件名中包含有 Text 的文件，但不包括“.”开头的文件
[ab-fm]*	当前目录下所有以 a、b、c、d、e、f 和 m 开头的文件名
[ab-fm]?	当前目录下所有以 a、b、c、d、e、f 和 m 开头且后面只跟一个字符的文件名
[!ab-fm]?或[^ab-fm]?	当前目录下所有不以 a、b、c、d、e、f 和 m 开头且后面只跟一个字符的文件名
/dev/sd?	目录/dev 下所有以 sd 开头且只有三个字符的文件名

需要说明的是，连字符“-”仅在方括号内且在中间时有效，表示字符范围，如在方括号外面或在方括号内最前或最后就成为普通字符了；字符“^”和“!”只在方括号内且位于开始位置才起“非”的作用，而“*”和“?”只在方括号外面是通配符，若出现在方括号内，它们也失去通配符的能力而成为普通字符了。例如，模式“-a[*?]abc”中只有一对方括号是通配符，“*”和“?”均为普通字符，因此，它匹配的字符串只能是“-a*abc”或“-a?abc”。

以上是通配符在文件名模式匹配中的应用，但它们在 shell 编程或不作为文件名模式匹配时还具有另外的意义（参见正则表达式）。总的来说，它们对于 shell 来说具有比较特殊的意义，因此在正常的文件名中不应使用这些字符。

3）注释符与注释

在所有编程语言中都有注释，以增加程序的可读性。在 shell 命令或 shell 编程中，规定从字符“#”开始以后到行末的部分作为注释，因此以“#”开头的行是注释行。以后将会发现，事情并非总是这样，例如，$#表示命令行参数的个数，而${#name}表示变量 name 的值作为字符串的长度。

若在行中需要使用“#”开始引导注释，则“#”前至少要有一个白空格。

4）转义字符

UNIX/Linux 系统中还有一个特殊字符“\”，用于对某些特殊字符的表示，基于此，字符“\”被称为转义字符（ESCape char）。部分特殊字符如表 3-2 所示。

表 3-2　Linux 的部分特殊字符

字　　符	意　　义	字　　符	意　　义	字　　符	意　　义
\a	响铃符	\x??	十六进制表示	\\	\
\b	退格键	\0???	八进制表示	\`	`
\f	换页	\t	Tab 键	\”	”
\n	换行	\v	垂直制表符	\e	ESC

2．特殊键

在 UNIX/Linux 系统中，键盘上常规键的用法与其他操作系统相同，但对于特殊键和组

合键，其意义就可能不同于其他操作系统。另外，特殊键在 UNIX/Linux 系统中的意义和使用方法可能还与所使用的终端类型有关。UNIX/Linux 系统的部分特殊键如表 3-3 所示。

表 3-3　UNIX/Linux 系统的部分特殊键

特　殊　键	意　　义
Ctrl+D	结束当前程序输入或结束当前程序，或从系统中注销
Ctrl+C	终止当前程序的执行
Ctrl+\	终止当前程序的执行（系统内部处理时不同于 Ctrl+C），可能产生内存映像文件
Ctrl+Z	挂起当前程序执行
Ctrl+S/Ctrl+Q	暂停/继续屏幕的输出滚动，必须配合使用
ScrolLock	暂停/继续屏幕的输出滚动，开关键
Ctrl+Alt+Del	默认动作为重启系统
Alt+F#	字符终端切换，如 Alt+F2 将终端切换至 F2
Ctrl+Alt+F#	从图形界面切换终端到字符终端，如 Ctrl+Alt+F1 从图形界面切换至 F1

3．保留字

任何编程语言或系统都有一定数量的保留字，bash 的部分保留字如下：

!，[，]，（，），{，}，break，continue，cd，echo，eval，exec，exit，export，function，getopts，hash，pwd，read，readonly，return，select，set，shift，test，time，trap，type，ulimit，unset，umask，wait，for，do，done，case，in，esac，if，else，elif，fi，while，until。

3.1.3　文件命名及文件类型

1．文件与文件名

操作系统都有文件的概念。文件是被命名的存储在某种介质（如磁盘、光盘和磁带等）上的一组信息的集合。UNIX/Linux 的文件均为无结构的字符流。文件名是文件的一种标识，一般情况下，它由字母、数字、下画线和圆点组成的字符串构成。其实，在 UNIX/Linux 系统中，可用于文件名的字符是广泛的，只要是用户能够输入的字符均可用于文件名，但在实际应用中应该使用具有实际意义的文件名。

Linux 支持长文件名，但要求文件名的长度限制在 255 个字符以内。为了便于管理和识别，用户可以把扩展名作为文件名的一部分，用圆点“.”区分文件名和扩展名。扩展名对于将文件分类是十分有用的。用户可能对某些大众已接纳的标准扩展名比较熟悉，如 C 语言程序以.c 为扩展名。用户也可以根据自己的需要定义自己的文件扩展名。

2．文件类型

UNIX/Linux 系统中有三种基本的文件类型：普通文件、目录文件和设备文件。

1）普通文件

普通文件是用于存放数据的文件，它是用户最经常面对的文件，又可分为文本文件和二进制文件。

（1）文本文件：以可阅读的 ASCII 码形式存储在计算机存储器中，大多数情况下，是以“行”为基本结构的一种信息组织和存储方式。

（2）二进制文件：是一般不能直接读懂的文件，只有通过相应的软件才能对其进行操作。二进制文件一般是经编译程序编译后生成的可执行程序、图形、图像或声音等。

2）目录文件

目录文件用于存储一组相关的文件项信息或文件说明信息，其中包括文件名及其属性的信息。在 UNIX/Linux 系统中目录只包括文件名和 i 节点号等相关信息，而文件的属性信息保存在 i 节点信息中。目录文件在形式上同普通文件一样，但具有目录属性，只能用目录管理命令来访问和管理。

目录文件中的文件项信息数据结构如下所示：

```
struct dir_entry {
        __u32       inode;                  //i 节点号，32 位无符号长整数
        … … … …
        unsigned char     name[255];        //文件名，最长为 255
};
```

当文件系统对文件进行搜索时，首先按名存取，查到文件名后，再由文件名得到对应的 i 节点号，最后通过 i 节点号找到文件属性和位置信息，从而实现对文件的访问。

一个目录文件内至少要有两个文件项：“.”—当前目录；“..” —上级目录。

目录文件的大小只能增加，尽管可以从目录中删除文件或子目录，但不能使目录变小。使目录变小的方法是删除后重建。

目录文件简称为目录或子目录，在图形界面下称为文件夹。因为目录也是文件，所以它的命名规则与文件命名规则相同。

3）设备文件

设备文件是 UNIX/Linux 系统的一个重要特色。UNIX/Linux 系统把每一个 I/O 设备都看成一个文件，与普通文件一样处理，这样可以使文件与设备的操作尽可能统一。从用户的角度来看，对 I/O 设备的使用和对一般文件的使用一样，不必了解 I/O 设备的细节。

设备文件与普通文件和目录文件不同，它除了在目录文件和文件说明信息表（i 节点）中占据相应的位置之外，并不占有实际的物理存储空间。当用户使用设备文件时，可通过设备的名称得到其 i 节点，然后通过其中的主、次设备号取得与内核中设备驱动程序的联系，从而达到访问设备的目的。

为了区分普通文件、目录文件和不同设备文件之间的不同，系统把设备文件称为“设备特别文件”，简称为设备文件。通常情况下，设备文件存放在系统的设备目录/dev 下。常见设备文件有以下几种。

（1）块设备文件（b）：以块方式存取的设备，如硬盘、磁盘、磁带等。

（2）字符设备文件（c）：以字符方式存取的设备，如字符打印机、显示器等。

（3）符号链接（l）：用于通过此文件的内容指向它所链接的文件或资源。

（4）套接字文件（s）：套接字分为文件套接字和网络套接字。文件套接字存在于文件系统中，让用户以文件的方式访问网络连接。

（5）管道设备文件（p）：用于进程之间通信的先进先出（FIFO）“临时文件”。

管道设备有以下两类。

（1）无名管道：主要用于有直接继承关系的父子进程之间的通信。当创建无名管道的进程结束后，无名管道自动消失。

（2）命名管道：命令管道是一种特殊的设备文件，在文件系统里以文件形式的存在。由于是以文件形式存在的，使得有家族关系的进程间可使用其进行通信，不具家族关系的进程也使用其进行通信。在通信结束后，若不主动删除它，它不会自动消失，而仍以文件的形式存在于文件系统中。

文件链接也有如下两类。

（1）硬链接（-）：两个或多个文件名共用一个文件体，也就是说一个文件可以具有多个不同的名字，但它们具有相同的 i 节点号。

（2）符号链接（l）：与硬链接不同的是，它们不是同一个文件，各有自己的 i 节点，但可以通过符号链接文件的内容访问被链接文件。符号链接也叫软链接，在 Windows 系统中被称为快捷方式。

硬链接来自 UNIX 系统本身，不能实现跨越文件系统的链接。符号链接是在现代计算机应用基础上发展起来的，具有广泛的灵活性，可以跨文件系统链接，可以指向网络资源，甚至可以指向一个不存在的资源或位置。

3.1.4 目录结构与路径

1．目录与目录结构

在操作系统中有大量的文件，如何有效地组织与管理它们，并为用户提供一个使用方便的接口是文件系统的一大任务。UNIX/Linux 系统以文件目录的方式来组织和管理系统中的所有文件。所谓文件目录就是将所有文件项或文件的说明信息采用树型结构组织起来，即常说的树型目录。UNIX/Linux 系统中，整个文件系统只有一个“根”（root），然后在根上分“杈”（directory），任何一个分杈上都可以再分杈，也可以长出“叶子”。“根”和“杈”在 Linux 中被称为“目录”或“文件夹”，而“叶子”则是一个个文件。在 UNIX/Linux 系统中，文件系统的根目录用符号“/”表示。

UNIX/Linux 系统通过目录将系统中所有文件分级、分层组织在一起，形成了树型层次结构，它以根目录为起点，所有其他目录都由根目录派生而来。

实际上，各个目录节点“之下”都会有一些文件或子目录，并且系统在建立每一个目录时，都会自动为它创建两个目录文件，一个是“.”，代表该目录自己；另一个是“..”，代表该目录的父目录，通过它取得与上级目录的联系。对于根目录，“.”和“..”都代表其自己。

UNIX/Linux 提供有管理文件和目录的方便途径。用户可以方便地从一个目录切换到另一个目录，也可为自己创建文件或目录，或可以把一个目录下的内容移动或复制到另一个目录下，并且还可与系统中的其他用户共享目录或文件。用户还可以设置目录和文件的管理权限，以便进行访问控制。

Linux 是一个多用户系统，操作系统本身的程序或数据存放在以根目录开始的某些专用目录中，这些目录有时被称为系统目录。用户也可创建自己的目录用于存放自己的程序或数据。Linux 系统的目录结构如图 3-1 所示（需要说明是，不同版本或厂家的系统中，目录结构可能存在差异）。

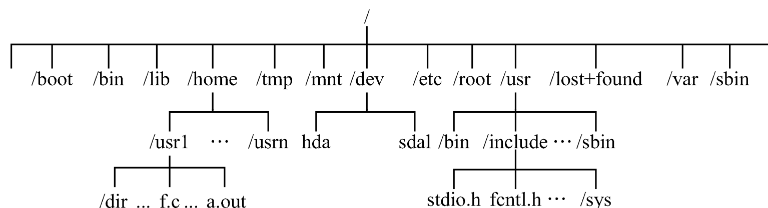


图 3-1　Linux 系统的目录结构

/：系统的根目录。

/dev：系统的设备目录，其中存放着几乎所有的设备文件。

/etc：存放 Linux 系统和大部分应用软件的配置文件。

/home：用户家目录所在的目录。家目录也称主目录。默认情况下，每创建一个用户，就会在这里新建一个与用户同名的目录，给该用户一个自己的空间，如/home/lisi 为用户 lisi 的家目录。

/root：root 用户的家目录。

/lost+found：丢失文件的存放目录。当系统因非法关机等原因造成文件系统损坏时，经修复后一些丢失的文件存放在这里。

/mnt：外部设备的挂装点，用于挂载设备文件。在系统安装后，可用于安装外来文件系统。用户也可以在此目录下创建自己的目录，如 cdrom、usb 和 hd 等用于安装光盘、U 盘、硬盘分区或其他设备。

/boot：Linux 的启动目录。系统内核文件，引导器 Grub 等存放在这里。

/proc，/sys：伪文件系统目录（参见 5.3.1）。

/usr：用户级目录。

/tmp，/usr/tmp：临时目录。

/sbin，/usr/sbin：系统级的命令与工具目录。

/bin，/usr/bin：用户级的命令与工具目录。

/usr/include：C 语言，内核编译所需要的头文件存放目录。

/lib，/usr/lib：库文件存放目录，其中有表态库和动态库。

/usr/src：Linux 源代码目录，编译内核时使用。

/var：通常用来存放一些经常变化的内容，如各种网络服务的工作目录都在这里。

/var/log：系统日志目录。

/var/spool：存放邮件、新闻、打印队列任务等。

/usr/share：存放共享使用的，如各种共享或在线帮助等。

/media、/run/media：桌面系统的新增目录，用于安装移动存储设备。

2．工作目录、用户主目录与路径

1）工作目录与用户主目录

自从用户登录到系统中后，每时每刻都“处在”某个目录之中，此目录被称为工作目录或当前目录（Working Directory）。工作目录用“.”表示，且可根据需要而随时改变。

默认情况下，用户登录到系统之初，其主目录或家目录（Home Directory）就成为其工作目录。每个用户都有自己的主目录，不同用户的家目录一般互不相同。root 用户的主目录

为/root，其他用户的家目录是在/home 下的与登录名相同的目录。

用户可以通过一个“~”字符或环境变量 HOME 来引用自己的家目录。

2）路径

路径是指从树型目录中的某个目录到另一个文件或目录的路线。路径表示符是由它所经过的目录名，中间用“/”链接构成的。

任一文件在文件系统中的位置可以由相对路径或绝对路径来决定。绝对路径是指从“根”开始的路径；相对路径是从用户工作目录开始的路径。应该注意的是，在引入链接的文件系统中，从某位置到另一位置的路径可能不只一个，因此又有物理路径和逻辑路径之分。绝对路径是确定不变的，而相对路径则随着用户工作目录的变化而不断变化。

当用户要访问一个文件时，可以通过路径名来实现，并且可以根据要访问的文件与用户工作目录的相对位置来引用它。例如，处在图 3-1 中 dir 目录的用户要访问目录 usrn 中的文件 file1，可通过相对路径../../usrn/或绝对路径/home/usrn/来实现，此时带有路径的文件名可表示为../../usrn/file1 或/home/usrn/file1，这种带有路径的文件名通常也称为文件名。

3.1.5　shell 命令格式与命令编辑

用户登录到系统后，可以看到一个 shell 提示符，root 用户的提示符为“#”，普通用户的提示符为“$”。用户可以在提示符后面输入任何命令及参数。例如：

$ date

用户登录成功后，实际上进入了 shell，它遵循一定的语法将输入的命令加以解释并传给系统。命令行中输入的第一个字必须是一个命令的名字，若有，则以后内容为由分隔符分隔的选项或参数，格式如下：

```
命令名 [选项 … ] [参数 …]
```

1．分隔符

在 Linux 系统中默认的分隔符为空格。

2．选项和参数

参数是命令操作的对象，而选项用于影响命令对对象的操作行为。选项是由符号“-”引导的字符或字符串，“-”是必需的，Linux 用它来区别选项和参数。在 Linux 系统中选项有两种形式：一种是传统 UNIX 风格的选项，以“-”开始，紧跟一个字符；另一种是 GNU 风格的选项，以“--”开始，紧跟着完整的英文单词或由“-”连接的单词组合。考虑到书写等因素，本书更多的是介绍传统 UNIX 风格的选项格式，只有在必要时才采用 GNU 风格的选项。两种风格的选项使用示例如下所示。

```
$ date                                  #以默认方式显示当前日期和时间
##以 Greenwich Mean Time 显示当前日期和时间
$ date -u                               #传统 UNIX 风格选项
$ date --utc                            #GNU 风格选项
$ date --universal                      #同上
##显示文件/etc/passwd 的属性信息
```

```
$ ls -l /etc/passwd                          #默认模式或-u。传统 UNIX 风格选项
$ ls -l -n /etc/passwd                       #传统 UNIX 风格选项
$ ls -l --numeric-uid-gid /etc/passwd        #GNU 风格选项
$ ls -l -n --time-style=iso /etc/passwd      #传统 UNIX 和 GNU 风格选项混合使用
```

由上可见，选项-u 影响了 date 的输出，-n 等选项也影响了 ls 的输出。

由于 Linux 命令一般都带有多个参数或选项，描述不便，本书在以后的描述中，在不引起混淆的情况下不再区分参数和选项，统称为参数。

3．命令行编辑特性

1）命令和文件名扩展

bash 命令行具有命令和文件名扩展特性。当输入一个还没完成的命令或文件名时，只需按“Tab”键就能激活命令和文件名扩展特性，从而“自动”完成该命令的剩余输入。如果有多个命令或文件的前缀相同，bash 将响铃并等待用户输入足够的字符，以便选择唯一的命令或文件名。如果找到，系统将自动补齐搜索到的命令或文件名，用户按回车键后，系统将执行这条指令。例如，若此时系统中只有一个以“pre”开头的文件 preface，则当用户输入 cat pre 后再按 Tab 键将补全整个内容。

```
$ cat pre              #按 Tab 键后变为：
$ cat preface
```

当备选文件名不唯一时，按下 Tab 键后，bash 会在补齐最大可能长度后响铃，若此时用户先后按下“Esc”和“?”，或两次按下 Esc，bash 将列出所有与输入的字符串相匹配的文件名供用户选择，然后用户在输入少量可以区分文件名的字符后，再按 Tab 键补全命令行。如果需要补全的文件名很长且区分度不高，则可重复使用以上方法，直到最终补全命令行为止。

设当前目录内有 decrease.c、deveice.txt、document 和 docudrama 文件，我们要显示 document 文件的内容，可按以下方法进行。

```
$ cat do<Tab>            #按 Tab 键后，响铃，结果如下：
$ cat docu<Tab>          #按两次 Tab 键后，或先后按 Esc 和?，显示如下可选内容：
docudrama document
$ cat docum              #输入 m，再按 Tab 键补齐
$ cat document           #回车执行命令
```

对于命令本身也可按同样方法扩展。

可以看到，bash 总是尽力根据用户输入的信息来补全命令。当无法根据现有信息补全命令时，则提示用户再给出进一步的信息，然后再根据用户的提示来进一步补全命令。

2）命令行编辑

bash 允许用户对正在输入的命令行进行编辑，编辑过程主要通过如表 3-4 所示的编辑键来完成。

3）历史记录

命令行实际上是可以编辑的一个文本缓冲区，在按回车键之前，可以对输入的文本进行编辑。在 bash 中，回车后命令行被保留在历史记录文件中，可通过光标键“↑”和“↓”在历

史记录中漫步，对出现的“当前”记录可以进行编辑和使用。利用这一功能可以重复执行以前执行过的命令，而无须重新输入该命令。

表 3-4　bash 的命令行编辑键

编　辑　键	意　　义	编　辑　键	意　　义
Ctrl+A	光标移至行首	Ctrl+K	删除光标处至行末内容
Ctrl+E	光标移至行尾	Ctrl+U	删除光标处至行首内容
Alt+F	光标前移至右一词末	Alt+D	删除光标处至单词末内容
Alt+B	光标后移至左一词前	Alt+DEL	删除光标处至单词首内容
Ctrl+L	清屏并在屏幕顶端重显当前行	Ctrl+W	删除光标处左一词或至词首内容
Ctrl+-	恢复上一次的操作	Ctrl+Y	将刚删除的内容插入到光标处

在 Linux 系统中，为每个用户在~/.bash_history 文件内预设了 1000 条记录的历史，并为每条记录一个编号。使用命令：

```
history
```

会列出的所有历史记录。若用户要重新执行某条历史记录的命令则可以使用以下格式：

```
!记录编号
```

需要说明的是，就某一个历史命令来讲“记录编号”是变化的，按以上格式引用的“记录编号”必须是上次刚刚执行过的 history 命令显示的编号，否则可能会执行错误命令。

用户还可使用格式：

```
history n                             #n 为数字
```

让其显示最近的 n 条历史记录。history 命令的使用示例如下。

```
#history                              #显示所有历史记录
#history 20                       #显示最近 20 条历史记录
  99   ls –l /
  … … … … … …
#!99                                  #执行第 99 条历史记录：ls –l /
ls –l /
```

3.1.6　标准流与输入/输出重定向

1．标准流

当执行一个命令时，shell 通常会自动为其打开三个标准流（文件）：标准输入流、标准输出流和标准错误流。三种标准流所使用的文件描述符及物理设备如表 3-5 所示。默认情况下，程序将从标准输入流读取输入数据，而将正常输出数据输出到标准输出流，将错误信息送到标准错误流。

表 3-5　标准 I/O 流及相关设备信息

标　准　流	shell 文件号	C 描述符	默认使用设备
标准输入	0	Stdin	键盘
标准输出	1	Stdout	屏幕
标准错误	2	Stderr	屏幕

直接使用标准流存在以下问题：

（1）输入数据从终端输入时，用户输入的数据只能用一次，易出错且修改不便。

（2）输出到终端屏幕上的信息只能看，不便使用。

UNIX/Linux 系统的标准 I/O 重定向可以解决这些问题。

2．I/O 重定向

I/O 重定向（也称作改道）是指通过文件的形式实现标准 I/O 流。I/O 重定向可以通过以下符号实现。

（1）<：标准输入重定向。

（2）>：覆盖方式标准输出重定向。若文件不存在则创建之，否则覆盖之。

（3）>>：追加方式标准输出重定向。若文件不存在则创建之，否则在其尾部追加。

（4）&>：标准输出和标准错误同时重定向。

（5）&>>：标准输出和标准错误同时重定向追加。

1）标准输入重定向

标准输入重定向是指让命令（或可执行程序）的标准输入从指定的文件中读取，也就是说，输入可以不来自键盘，而来自一个指定的文件。因此，标准输入重定向主要用于解决一个命令的输入源数据，尤其是输入数据量大或重复使用问题。

例如，命令 wc 可用于统计指定文件包含的行数、单词数和字符数。

```
#wc /etc/passwd          #统计文件/etc/passwd 的行、单词和字符信息。也可使用：
#wc < /etc/passwd        #标准输入重定向
```

2）标准输出重定向

标准输出重定向是指把命令（或可执行程序）的标准输出保存到文件中，可用两种方式：覆盖方式(>)和追加方式(>>)。例如：

```
#ls > /tmp/dir.out          #覆盖方式，将当前目录信息重定向到文件/tmp/dir.out
#ls /usr >> /tmp/dir.out    #追加方式，将/usr 信息重定向追加到/tmp/dir.out
```

标准输出的重定向是有副作用的，一是它可能覆盖已经存在的文件，二是它可能会在一个具有完整意义的文件后面追加一些外来或无意义的信息。

3）标准错误的重定向

标准错误的重定向与标准输出的重定向意义相同，但它是针对标准错误的。在对标准输入和标准输出重定向时我们没有使用文件号，但在对标准错误改道时要使用文件号 2。方法是：

```
命令 2> 文件                    或
```

```
命令 2>> 文件
```

在 2 和>之间不能有空格。例如：

```
$ ls -l /home/w-w-w 2> /tmp/err.out              #将标准错误重定向到文件/tmp/err.out
$ cat /home/w-w-w 2>> /tmp/err.out               #将标准错误追加到文件/tmp/err.out
$ cat /tmp/err.out                               #查看文件/tmp/err.out 的内容
```

4）标准输出和标准错误同时重定向

使用符号&>或&>>可以让一个程序的标准输出和标准错误同时重定向到文件，前者用于覆盖方式，后者用于追加方式。示例如下：

```
$ ls -l -d /    /home/w-w-w &> /tmp/err.out      #将标准输出和标准错误同时重定向到/tmp/err.out
$ ls -l -d /boot /home/w-w-w &>> /tmp/err.out    #将标准输出和标准错误同时追加到/tmp/err.out
$ cat /tmp/err.out                               #查看文件/tmp/err.out 的内容
```

3.1.7 管道与三通

管道机制是在两个或多个进程之间建立一种连接，使得前一个命令的输出作为后一个命令的输入。管道机制常用于进程间的通信。将一个程序或命令的输出作为另一个程序或命令的输入，有两种方法：

（1）通过临时文件将多个命令或程序结合在一起。

（2）通过管道机制。

后者优于前者。

1．管道

实现管道机制的符号是“|”，其方法为：

```
命令 1 |  命令 2 | … |命令 n
```

例如：

```
#ls -l /dev | wc -l                              #统计系统设备目录/dev 内有多少文件或子目录
#cat /etc/passwd | grep "root:"|wc -l            #统计/etc/passwd 内有多少行包含 root:
```

2．三通（tee）

有时候，对某文件处理时，既要在屏幕上看到输出，又要同时保存一个副本，这时可使用管道与 tee 命令配合来实现三通。

tee 命令的功能是读取标准输入的数据，并将其内容同时输出到标准输出和指定文件，其用法为：

```
tee [-ai] [files]
```

选项-a 指定使用追加方式，若不指定则采用覆盖方式；-i 用于忽略中断信号（^C）。tee 命令的使用示例如下：

```
#cal | tee calendar       #将当月日历表显示到屏幕的同时也保存到文件 calendar 中
#ls / | tee f1 f2         #将根目录短式列表显示到屏幕上的同时也保存到文件 f1 和 f2 中
```

3.1.8　环境变量与变量

1．变量

在 UNIX/Linux 系统中，用户可以定义自己的变量。定义后的变量就像在其他程序设计语言里一样被引用。引用变量时需要使用“$”作为变量名的前导字符。

变量的定义方法：

```
var_name=var_value
```

变量的定义和使用示例如下：

```
$ x=123                    #定义一个数值变量 x，其值为 123
$ y='I am a student'       #定义一个字符串变量 y，其值为'I am a student'
$ echo $x $y $HOME         #显示用户变量 x，y 和环境变量 HOME 的值
$ z= " $x $y "             #由变量 x 和 y 定义变量 z，其值为'123 I am a student'
```

由于 shell 是解释语言，所以在变量定义和使用时不需要声明类型，shell 在使用变量时会根据变量的值来具体确定变量的类型。

2．环境变量

shell 在开始执行时就已经定义了一些和系统工作环境有关的变量，用来控制用户程序的执行，或为用户程序的执行提供环境支持。环境变量可用命令 env、export 或 set 来查询，比如：

```
$ export        #显示所有的环境变量
$ env           #同上
```

常用的 shell 环境变量如下。

（1）HOME：家目录。默认情况下，用户成功登录后所在目录的完全路径名。

（2）LOGNAME：登录用户名。用户成功登录时使用的用户名。

（3）IFS：命令行内部参数、选项间的分隔符。默认为空格。

（4）PATH：由冒号分隔的目录路径名。shell 将按 PATH 变量中给出的路径顺序搜索键盘输入的命令名，找到的第一个与命令名称一致的可执行文件将被执行。

（5）TERM：终端的类型。常用的有 Linux 和 xterm 等。

（6）PWD：当前工作目录的绝对路径名。该变量的取值随 cd 命令而变化。

（7）OLDPWD：旧目录。刚刚离开的目录。该变量的取值随 cd 命令而变化。

（8）PS1：主提示符。默认情况下，超级用户的主提示符是#，普通用户的主提示符是$。

（9）PS2：辅提示符（继续行用提示符）。在 shell 接收用户输入命令的过程中，如果用户在输入行的末尾输入“\”然后回车，或者当用户按回车键时 shell 判断出用户输入的命令没有结束时（如引号或括号不成对等），就显示这个辅助提示符，提示用户继续输入命令的其余部分，默认的辅助提示符是“>”。

用户还可定义自己的环境变量（见后述）。

3.1.9 引号机制、命令替换与变量替换

1．引号机制

在 shell 中有三种引号：单引号（'），双引号（ " ）和反单引号（`），前两者用于变量定义，后者用于命令替换。

1）单引号

由单引号括起来的字符都作为普通字符，特殊字符用单引号括起来以后也会失去原有意义。例如：

```
#string='$PATH'          #定义 string 变量，其值为'$PATH'
#echo $string            #显示变量的值：$PATH
```

2）反单引号

反单引号的作用是命令替换。所谓命令替换是指反单引号内的内容将作为命令首先被执行，然后再将命令执行的标准输出替换反单引号及其括号位置的信息。例如：

```
#x=`pwd`                 #通过命令 pwd 替换定义变量 x。pwd 的功能是显示用户工作目录
#y=`whoami`              #通过命令 whoami 替换定义变量 y。whoami 的功能是显示用户名
#echo $x $y              #显示变量 x 和 y 的内容
```

命令替换的另一种形式是：

```
$(cmd)
```

它的作用与`cmd`相同。例如：

```
#MyVar=$(whoami)         #通过命令 whoami 替换定义变量 MyVar
#echo $MyVar $(pwd)      #显示变量 MyVar 和$(pwd)的值
```

命令替换也叫命令扩展。

3）双引号

双引号的作用与单引号的功能基本一样，可用来定义变量，所不同的是在双引号内可进行变量替换和命令替换。双引号中的特殊字符仍具有特殊意义。所谓变量替换，是指在双引号中的对变量的引用将被替换成变量的值，变量替换也叫变量扩展。若在双引号内使用特殊字符且又把它作为特殊字符本身来看待，则必须使用转义字符，比如，双引号中的双引号必须表示为\”，\必须表示为\\。

双引号使用示例如下：

```
$ myname=`whoami`               #通过命令替换定义变量 myname
##定义变量 myself。注意使用了变量替换和命令替换
$ myself="I am a student, my uname is $myname and my work dir is `pwd`."
##定义变量 myself1。注意使用了继续行和转义字符
$ myself1="I am a student my uname is \"$myname\" and my work dir is \"`pwd`\""
$ echo $myself                  #显示变量 myself 的值
$ echo $myself1                 #显示变量 myself1 的值
```

2．参数替换

参数替换主要是指命令行参数中的变量和命令替换。在执行命令时，命令的命令行参数既可以是常量也可以是变量，也可以包含命令替换。设当前目录下有命令 DispAllVar 用于显示它自己的所有命令行参数，变量 x 的值为“test parm”，则命令

```
#./DispAllVar I am `whoami` the value of x is $x
```

的输出为：

```
I am root the value of x is test parm
```

由此可见，在程序的执行过程中`whoami`被替换为 root，$x 被替换为“test parm”。像这样参数在命令行被替换的过程称为参数替换。甚至，还可以定义一个变量：

```
D=DispAllVar
```

之后，程序的执行可简写为：

```
#./$D I am `whoami` the value of x is $x
```

在这里，程序名也可以是变量。

说明：程序 DispAllVar 可以由如下命令产生：

```
#echo 'echo $*' > DispAllVar; chmod +x DispAllVar
```

3.1.10　shell 命令的执行

1．命令搜索

shell 拥有自己的内部命令，也能被系统中其他应用程序所调用，如改变工作目录命令 cd 是内部的；还有一些命令，如文件复制命令 cp 和文件移动命令 mv 等，是存在于文件系统中某个目录下的程序；也还有一些命令，如 echo，既是内部的也是外部的。对于一般用户而言，不必关心一个命令是内部的还是外部的。

默认情况下，当用户从键盘输入一个命令时，shell 首先检查它是否带有路径。若有路径则按路径搜索它，若找到则执行之，否则报错（告诉用户命令找不到或不存在）。若没有带路径，则首先检查它是否为内部命令，若是则执行之；否则，再检查是否是一个外部命令。此时，shell 在 PATH 指定的搜索路径里寻找它。如果能够成功找到，则执行之，执行完毕后将输出返回给用户；如果找不到，将会显示一条错误信息。

对于既是内部又是外部命令者，或者说内、外部命令同名者，shell 也有一种机制可以有选择地执行之，但默认情况下是执行内部命令。

如果键盘输入的命令是一个别名，则优先于内部命令。

2．标准流与程序 I/O

当一个 shell 命令执行时，系统首先为其打开三个标准流：标准输入、标准输出和标准错误。程序执行时，默认从标准输入读、向标准输出写、出错时将错误信息写向标准错误。

当一个程序需要从输入文件中读取、向输出文件输出信息时，若没有为之提供相关文件，则使用标准 I/O，将输入文件定向到标准输入（终端键盘），将输出文件定向到标准输出（终端显示器）。需要时，也可将标准输入和标准输出重定向到管道。当从文件读取信息时，结束符为文件结束标志 EOF；当从标准输入读取信息时，输入完成时的结束符为^D。例如，命令 grep 是用于信息过滤的，现在通过命令 ls /dev > x 产生一个文件 x，然后用于 grep 从 x 中过滤 SCSI 硬盘设备信息。方法是：

```
$ grep sd x            #使用文件
$ grep sd < x          #使用标准输入
```

如果仅仅是为了统计 ls /dev 输出中的信息，我们完全可以不使用中间文件 x，而直接使用管道：

```
$ ls /dev | grep sd    #使用管道
```

如果需要，也可将以上命令的输出重定向到输出文件或通过管道传给其他程序，例如：

```
$ ls /dev | grep sd > y     #通过标准输出定向到文件 y
$ ls /dev | grep sd | wc    #通过标准输出定向到管道
```

由于 UNIX/Linux 系统的 shell 命令大多做成了从标准输入读，向标准输出写的过滤器，用户可将从标准输入读的信息重定向到输入文件，将向标准输出写的信息重定向到输出文件，也可将标准错误重新定向到文件。

3．命令的返回值

UNIX/Linux 系统默认约定，shell 命令在结束时向调用者返回一个状态码或返回值，以表示执行成功与否。返回值为 0 表示成功，非 0 表示失败。当命令是由管道连接的命令串，或是命令组时，最后执行的命令的返回作为整个命令串的返回值。

返回值可以通过$?来访问。

如果系统中只有 SCSI 而没有 IDE 硬盘时，则命令

```
#ls /dev/sd*           #执行命令，显示系统中的硬盘设备
#echo $?               #显示返回状态（值）
```

将输出 0，表示/dev 目录内有 SCSI 硬盘设备，而命令

```
#ls /dev/ | grep '^hd'; echo $?
```

将显示非 0（此处为 1），表示系统中没有 IDE 设备。

3.1.11　shell 种类

UNIX/Linux 中的 shell 有多种类型，其中最常用的几种是 Bourne shell（sh 或 bsh）、Bourne again shell（bash）、C shell（csh）、tc shell（tcsh）、Korn shell（ksh）和 Z shell（zsh）等。

bsh 是 Stephen Bourne 在 Bell 实验室设计的 shell，是 UNIX shell 的一种，并且在每种 UNIX 上都可以使用，且在众多的 UNIX 系统中 bsh 基本保持一致，它是大多数 UNIX 系统默认的 shell。

bash 是 GNU 工程中使用的 shell，即 GNU 操作系统的默认 shell，bash 是在 bsh 基础上

发展起来。bash 与 bsh 稍有不同，它兼有 csh、tcsh 和 ksh 的特色。大多数 bsh 脚本程序可在其上不加修改就可运行，但 bash 的脚本程序则不一定能在 bsh 上运行。bash 实现了 IEEE POSIX 1003.2/ISO 9945.2 Shell 和工程规范。Linux 操作系统默认的 shell 是 bash。

csh 是在 bsh 之后 ksh 之前由 vi 的设计者 Bill Joy 编写的，它不是 bsh 的扩展。它采用 C 语言作为语法模型，是一种比 Bourne shell 更适于编程的 shell。Linux 为喜欢使用 C shell 的人提供了 tcsh。tcsh 是 C shell 的一个扩展版本。tcsh 包括命令行编辑、可编程单词补全、拼写校正、历史命令替换、作业控制和类似 C 语言的语法，它不仅和 Bash shell 提示符兼容，还提供比 Bash shell 更多的提示符参数。

ksh 是由 Bell 实验室的 David G. Korn 命名的，可以说它是对 bsh 的发展，在大部分内容上与 bsh 兼容，几乎所有的 bsh 脚本程序都可以在 ksh 上运行，但反之不行。Korn shell 集合了 C shell 和 Bourne shell 的优点。Linux 系统提供了 pdksh（ksh 的扩展），它支持任务控制，可以在命令行上挂起、后台执行、唤醒或终止程序。

Linux 也并没有冷落其他 shell 用户，还提供了一些流行的 shell，如 ash、zsh 等。每个 shell 都有它的用途，有些 shell 是有专利的，有些能从互联网上或其他来源获得。

3.2　Linux 系统的基本命令

Linux 系统中的 shell 命令非常多，本节主要介绍目录操作、文件操作、文本文件处理、进程管理、时间管理、文件或目录比较等基本操作命令，其他专用的管理将分散到各相关管理章节介绍。

3.2.1　目录基本操作命令

1．列目录内容（ls）

1）功能与用法

ls（list）的功能是列指定目录的内容，并将输出按某种方式排序。若不指定目录，则默认当前目录。其用法为：

```
ls [options] [files]
```

2）参数说明

Ls 命令的部分参数如表 3-6 所示。

表 3-6　ls 命令的部分参数

选　　项	意　　义
-a	列出所有文件及目录（Linux 规定以点“.”开头为隐藏文件，默认情况下不会列出）
-A	同-a，但不列出.和..
-b	以八进制数字方式列出不可显示的文件
-i	列出目录时显示 i 节点信息
-l	长格式列表

（续表）

选　　项	意　　义
-F	短格式列表，文件名称后加一符号表示属性：可执行文件加*，目录/
-r	将文件以相反排序方式列出（默认按文件名 ASCII 码值排序）
-R	递归列出目录及其子目录
-t	按文件依修改时间排序
-U	以不排序方式列目录

3）示例

```
#ls -ltr s*                          #列出当前目录下所有名字以 s 开头的文件，按修改时间倒排序
#ls -lR /bin                         #递归列/bin 目录及其子目录
#ls -ad .*                           #列所有文件名为.开头的文件
#ls -AF                              #列当前目录，在目录名后加/，可执行文件名后加*
#ls -lid /usr/bin/ls /dev/sda /etc   #以长格式列文件且列出文件和 I 节点号
```

ls 命令位于目录/bin，经常使用的命令 ls 是它的一个别名，系统中还有它的另一个别名 ll，相当于 ls -l。

4）ls 的输出说明

命令“ls　-lid　/usr/bin/ls /dev/sda /etc”输出如下：

```
  2039 b rw-rw----.   1 root disk      8, 0   Mar 8 09:06 /dev/sda
651522 d rwxr-xr-x. 141 root root     12288   Mar 8 09:08 /etc
   894 - rwxr-xr-x.   1 root root    141596   Nov 1 01:21 /usr/bin/ls
   ①   ②    ③         ④   ⑤    ⑥        ⑦           ⑧          ⑨
```

各域的意义是：①I 节点号；②文件类型；③权限；④链接数；⑤主；⑥组；⑦普通文件为长度，特别文件为主次设备号；⑧最后修改时间；⑨文件名。

2．建立目录（mkdir）

1）功能与用法

mkdir（make directory）命令的功能是创建子目录，一次可以创建一个或多个子目录，还可以一次创建包括全部父目录和子目录在内的一个完整的目录结构，它的使用受权限的制约。mkdir 的用法为：

```
mkdir [-p] [-m MODE] dirs
```

2）参数说明

-p：如果要创建的目录存在也不报错，则必要时可一同创建父目录。-m，--mode=MODE：按照权限 MODE 创建子目录（关于权限控制参见第 5 章）。

3）示例

```
#mkdir temp                     #在工作目录下，建立一个名为 temp 的子目录
#mkdir temp2 temp3 temp4        #一次创建多个目录
#mkdir –p temp5/child           #在子目录 temp5 下创建子目录 child，若父目录不存在
#则一同创建之。若不使用-p 参数，在父目录不存在时，将导致子目录创建失败
```

3．删除目录（rmdir）

rmdir（remove directory）命令的功能是删除空的子目录，其使用受权限制约。rmdir 的用法为：

```
rmdir [-p] dirs
```

使用-p 参数，rmdir 在删除一个目录时，若其子目录也是空的，则一并删除。rmdir 命令使用示例如下。

```
#rmdir temp                    #删除子目录 temp，若非空则报错
#rmdir -p a/b/c                #删除目录时一同删除其空子目录
##注：若 a 只有子目录 b，b 只有子目录 c，则以上命令等同于：
#rmdir a/b/c;rmdir a/b;rm a
```

4．改变工作目录（cd）

cd（change directory）命令是改变用户的当前工作目录。它的使用受权限制约，用户必须对要进入的目录有执行权。其用法为：

```
cd [dir]
```

它需要一个目的目录作为参数，若不指定，则默认切换到用户的家目录$HOME。使用示例如下：

```
#cd /tmp                       #切换到目录/tmp
#cd ~-                         #切换到刚离开的目录。相当于 cd $OLDPWD
#cd ..                         #切换到上级目录
#cd                            #切换到家目录$HOME。相当于 cd $HOME
```

5．显示当前目录（pwd）

pwd (print working directory)命令的功能是显示当前目录。其用法为：

```
pwd [-P] [-L]
```

选项-P、-L 分别用于显示当前目录的物理和逻辑位置，默认为后者。

3.2.2　文件操作基本命令

1．显示文件的内容或合并文件（cat）

1）功能及用法

cat（catenate）命令用来把文件内容显示到屏幕上，也可以同时显示多个文件。当不带参数运行 cat 时使用标准输入。使用 cat 可以通过 I/O 重定向的方法将多个文件的显示内容改道到某个文件中，从而实现文件合并。cat 的用法为：

```
cat [ options ] [ files ]
```

2）参数说明

cat 命令的部分选项如表 3-7 所示。

表 3-7　cat 命令的部分选项

选　　项	意　　义	选　　项	意　　义
-E	在行末显示$符号	-s	当有一个或多个空行时只显示一个
-n	为所有行添加行号	-T	将 Tab 键显示为^I(Ctrl+I)
-v	显示所有内容。除回车和 Tab 键外，不可显示的字符用^或 M-表示（^表示 Ctrl 键，M-表示 Alt 键）		

3）应用举例

```
#cat test.txt                    #显示文件 test.txt 的内容
#cat -n test.txt                 #显示文件 test.txt 的内容，并为所有行添加行号
#cat *.txt                       #显示所有的*.txt 文件
#cat tst1.txt tst2.txt > tst.txt #将 tst1.txt 和 tst2.txt 的内容合并为 tst.txt
#cat > myfile                    #从键盘读入内容并将其保存到文件 myfile。按^D 结束
```

2．分屏显示文件内容（more）

1）功能及用法

more 命令的功能是分屏显示文件的内容。more 命令是 UNIX/Linux 操作系统中分页命令家族中的一员。分页命令能使用户在浏览文件的时候可以一次一屏或一行进行，这在阅读大文件时特别有用。more 命令的用法为：

```
more [-dflpcsu] -lines [+linenum | +/pattern] files
```

2）参数与命令说明

more 命令的参数如表 3-8 所示，命令如表 3-9 所示。

表 3-8　more 命令的部分参数

参　　数	意　　义
+/pattern	从第一个正规表达式 pattern 处开始显示（参见第 10 章）
+linenum	从第 linenum 行开始显示
-lines	指定满屏行数为 lines，默认时为 24 行
-d	屏幕底端的默认提示。默认提示为--more—(##%)
-l	忽略^L（换页）。默认时遇到^L 时会暂停
-s	压缩多个空行为一行

表 3-9　more 命令的部分屏幕操作命令

命　　令	功　　能	命　　令	功　　能
Q/q/^C	退出	!cmd	执行 shell 命令
=	显示当前行行号	:n	下一个文件
#d/#b	向前/后滚动#行（#为数字）	:p	上一个文件
空格	向前翻一屏	:f	显示当前文件名和行号
回车	向前翻一行	^L	重显本屏内容
?/h	帮助	.	重复上一个命令

3）应用举例

```
#more /etc/passwd                #分屏显示/etc/passwd 的内容
#ls -l /dev | more               #列出系统设备目录并通过 more 分屏显示
```

用户可使用表 3-9 所示的命令键对显示过程进行控制，可通过按“q”键或在文件结束时退出。

3．使用 less 命令浏览文件

less 命令和 more 一样都是分页处理命令，但 less 比 more 的功能更强大。它改进了 more 命令中的一些功能，并又添加了许多其他特色，例如，可以使用 PgUp 和 PgDn 翻页，使用 HOME 和 END 直接跳到文件开始或结尾等。

less 命令是许多程序（如 man）等使用的默认的分页处理命令。如果想阅读压缩文件（带有.gz 或.zip 等后缀的文件），它会自动调用 less 命令来处理。通过 less 分屏显示/dev/目录内容的示例如下：

```
ls -l /dev | less
```

4．文件或目录的删除（rm, unlink）

1）功能及用法

rm（remove）命令的功能是删除文件。它可删除各种类型的文件，这并不奇怪，因为 UNIX/Linux 系统把设备和目录等都作为文件看待。由于它在删除文件时表现的功能特别强大，且 UNIX/Linux 系统内被删除的文件是不可恢复的，所以使用时要特别小心。rm 命令的使用方式为：

```
rm [options] files
```

2）参数说明

rm 命令的部分参数如表 3-10 所示。

表 3-10　rm 命令的部分参数

选　　项	功 能 描 述
-i	交互方式。删除前逐一询问确认，回答时 y/yes 表示确认，其他为放弃
-f	强制方式。强迫删除，删除时不进行提示
-r/-R	递归方式。删除目录时，连同其子目录一起删除。删除目录时必须选此项
--	删除以-开头的文件

Linux 系统还有一个命令 unlink 用于删除单个文件，但不能删除目录。

3）示例

```
#rm   myfile   hisfile           #交互式删除，删除前需要用户确认
#rm   -r   -f   myf.c mydir      #同时删除文件 myf.c 和目录 mydir
#rm   --   -xyz                  #删除具有特殊名字的文件-xyz
#rm   \*   \\                    #删除名字为*和\的文件
#unlink /tmp/x                   #删除/tmp/x
```

5．文件移动或更名（mv）

1）功能及用法

mv（move）命令的功能是文件位置移动或更名。其用法为：

```
mv [options] source    dest
mv [options] source ... directory
```

第一种用法是将源文件移动到目的文件，可用于文件移动或更名；第二种用法是将一批文件移动到某个目录 directory。mv 命令是有副作用的，但可使用备份方式避免。

mv 至少需要两个参数，一个用于源文件，另一个是目标文件或位置，若目标为当前目录可用“.”表示，但不能省略。

2）参数说明

mv 命令的部分选项如表 3-11 所示。

表 3-11　mv 命令的部分选项

选　　项	功 能 描 述
-b	若需覆盖，则在覆盖前备份。行为受-S 和-V 参数的影响，默认时在文件名后追加~
-f	参见 rm
-i	参见 rm
-u	若目标文件已经存在，则比现文件旧时覆盖，否则不覆盖
-S suff	定义备份时使用的后缀为 suff，默认为~，与-b 配合使用
-V=<method>	version-control 控制。method 可取值为：none/off：不备份；never/simple：按预设方式备份；t/numberd：文件名尾部增加备份次数，如~1~,~2~；nil/existing：若已有备份方式则使用原备份方式

3）示例

```
#mv file1 file2                     #若 file2 存在，则在不使用备份时将被覆盖
#mv myfile /tmp/tmpf                #将文件 myfile 移动至/tmp/tmpf 下或更名为/tmp/tmpf
#mv file1 file2 mydir               #将文件 fiel1，file2 移动到目录 mydir 下
#mv -bf –V"t" file1 file2 mydir     #以备份方式移动文件
```

6．文件和目录复制（cp）

1）功能及用法

cp（copy）命令的功能是文件或目录复制。不能复制设备文件，但却可复制设备文件的内容以构造映像。其用法为：

```
cp [options] source    dest
cp [options] source ... directory
```

第一种用法是将源文件复制到目标文件，第二种用法是将一批文件复制到某个目录 directory。cp 命令也是有副作用的，但同样可使用备份方式加以避免。

cp 至少需要两个参数，一个用于源文件，另一个是目标文件或位置，若目标为当前目录可用“.”表示，但不能省略。

2）参数及说明

表 3-11 所示的 mv 命令的参数在 cp 命令中均可使用，cp 命令不同于 mv 的部分选项如表 3-12 所示。

表 3-12　cp 命令的部分选项

选　　项	功 能 描 述
-d	复制时复制符号链接，若不指定则复制符号链接指向的文件
-p	复制源文件时，保留源文件的属性信息
-l	不作复制，只创建硬链接
-s	不作复制，只创建符号链接
-R/-r	递归复制。复制目录时，一同复制其中的所有文件和子目录

3）示例

```
#cp file1 file2                  #将文件 file1 复制到 file2，默认采用交互方式
#cp /tmp/myp.c .                 #将文件/tmp/myp.c 复制到当前目录
#cp *.c *.txt /tmp               #将文件*.c 和*.txt 复制到目录，保持原文件名
#cp file1 /tmp/file2             #将文件 file1 复制到/tmp 并更名为 file2
#cp –rp dir1 dir2                #将目录 dir1 复制到目录 dir2，且保持原属性
#cp –R file1 file2 dir1 dir2     #将文件 file?和 dir1 复制到目录 dir2
#cp /dev/cdrom /tmp/disk.img     #构造光盘映像到文件/tmp/disk.img
```

7．显示文件的开始或结尾部分（head/tail）

head 和 tail 命令可用于显示文件开始和结束部分，用户可以通过参数-num 或-n num 来决定要显示的行数，默认为 10 行。其基本用法为：

```
head [-num | -n num ] [ files ]
tail [-num | -n num ] [ files ]
```

head 和 tail 均可处理多个文件，此时输出中每个文件的内容均以

```
==>文件名<==
```

分隔，若要抑制文件名输出可以使用-q 参数。

8．文件的格式输出（pr）

1）功能及用法

pr（print）命令用于对文本文件进行加工并按指定格式输出。pr 可对文本文件按分页、分栏要求进行加工，若不指定输入和输出文件，则使用标准 I/O。在 pr 的默认输出中，页面长度为标准的 66 行，每页都有包括 56 行的正文，前、后各保留 5 行作为页眉和页脚。在页眉中有包含打印日期、时间、文件名和页码的标题。用户可能通过命令行参数来控制 pr 的行为。pr 命令的用法为：

```
pr [ options ] files
```

2）参数说明

pr 命令的部分参数如表 3-13 所示。

表 3-13　pr 命令的部分参数

选　　项	功 能 描 述
-#	指定分栏数。#为数字，默认为 1
-a	分栏时使用交叉方式（即按先后顺序交叉分栏），而非对分方式
-c	显示控制字符。显示方式为^x 或\###
-F/-f	使用^L 换页，而不以填充空行方式
-h string	使用指定的头字符串 string 作为页眉，而非默认方式
-J	取消固定列宽，将各栏合并为一长整行
-l #	指定页长为#行（默认为 66 行）
-m	合并文件。多个文件并列输出，每个占一栏
-n	在行前输出行号
-sCHAR	指定分栏字符串 CHAR，默认为 Tab
-SSTRING	指定分栏字符串 STRING。若不指定时默认为 Tab，-J 时为一个空格
-t	取消页眉和页脚
-w #	指定页宽为#个字符，用于多栏输出
-W #	指定页宽为#，除非与-J 配合使用，否则超宽部分被截掉

3）示例

```
#pr -n prog.c              #给 C 程序 prog.c 添加行号后输出
#pr -2 mytxt.txt           #以对分方式分 2 栏输出文件 mytxt.txt
#ls /dev | pr -a -3        #以交叉方式分 3 栏显示/dev 中的内容
#pr -n –f prog.c > x       #给 C 程序 prog.c 添加行号且以\f(^L)分页后存放到 x
```

9．以指定格式或进制显示文件内容（od）

1）功能及用法

od（octal dump）命令的功能是按不同进制或格式显示文件内容，默认为八进制。od 可以处理多个文件，此时输出将文件按排列顺序合并。若不指定输入文件或输入文件为-，则使用标准输入。其用法为：

```
od [options] files
od --traditional [files] [[+]offset [[+]label]]
```

2）参数说明

od 命令的部分参数如表 3-14 所示。

表 3-14　od 命令的部分参数

选　　项	功 能 描 述
-A type	地址类型：d,o,x 和 n 分别代表十进制、八进制、十六进制和无地址，默认为十六进制
-j<#>	跳过#个字符

（续表）

选　　项	功 能 描 述
-N<#>	只读入#个字符
-t<type>	设置输出格式。type:a，c，d#，f#，o#，u#，x#分别为命名字符、字符、整数、浮点数、八进制、无符号整数和十六进制数，#为数字表示位数
-w<#>	设置每列的最大字符数为#
--traditional	接收传统格式。传统格式可为：-a(-ta), -b(-toC), -c(-tc), -d(-tu2), -f(-tfF), -h(-tx2), -i(-td2), -l(-td4), -o(-to2), -x(-tx2)
-v	输出时不省略重复的数据

3）示例

```
#od -t x1 x              #按十六进制方式显示文件 x
#od -N32 -ta a.out       #按命名字符显示 a.out 前 32 个字符
#od -N32 -tx1c a.out     #按 1 位十六进制和字符方式分别显示 a.out 前 32 个字符
```

3.2.3　文件属性基本操作

1．确定文件类型(file)

1）功能与用法

命令 file 用于确定文件的类型，其用法为：

```
file [-bciknsvzL] [-f namefile] [-m magicfiles] file …
file -C [-m magicfiles]
```

2）参数说明

file 命令的部分参数如表 3-15 所示。

表 3-15　file 命令的部分参数

参　　数	功　　能
-b	输出结果时不显示前导文件名
-f file	从文件 file 中读取要确定类型的文件名
-i	显示 MIME 类型的输出，而非传统文本
-L	跟踪符号链接，显示链接目标的文件类型
-m filelist	指定幻数文件（filelist），而非默认的/usr/share/magic
-s	用于对特别文件进行进一步分析，以报告其上文件系统的类型
-z	尝试读取压缩文件的内容

3）示例

```
#file /bin/bash  /etc/inittab  /lib/libc.*     #确定文件类型
#file -s /dev/sda1                             #确定特别文件及其上的文件系统类型
```

2．显示文件或文件系统状态信息（stat）

1）功能及用法

stat 命令用于显示指定文件的状态信息或其所在文件系统的状态信息——属性信息。属性信息包括文件名、文件大小、文件类型、占用磁盘块数、i 节点号、链接数、存取权限、UID、GID 和三个时间。其用法为：

```
stat [options] files
```

2）参数及说明

stat 命令的部分参数如表 3-16 所示。

表 3-16　stat 命令的部分参数

参　　数	意　　义	参　　数	意　　义
-f	显示文件所在文件系统的信息	-L	跟踪符号链接
-cFORMAT	指定显示格式	-t	精简模式

说明：对于-cFORMAT，可以使用的格式参数如下。

%A/%a：权限；%b：已经分配的块数；%D/%d：设备号；%F：文件类型；%G/%g：文件的组；%h：硬链接数；%i：i 节点号；%N/%n：文件名；%o：I/O 大小；%s：大小；%T/%t：主/次设备号；%U/%u：文件所有者；%X/%x：最近访问时间；%Y/%y：最近修改时间；%Z/%z：最近改变时间。X、Y 和 Z 为从 1970.1.1 0:0:0 至今所经历的秒数。

3）示例

```
#stat /usr/bin/wc                    #显示文件/usr/bin/wc 的状态信息
#stat -f /usr/bin/wc                 #显示文件/usr/bin/wc 所在文件系统的状态信息
#stat /dev/cdrom                     #显示文件/dev/cdrom 的状态信息
#stat -L /dev/cdrom                  #显示文件/dev/cdrom 的链接文件状态信息
#stat -c"%A/%a %h %s" /etc/hosts     #显示/etc/hosts 的权限、链接数和长度
```

4）属性说明

命令 stat /usr/bin/wc 的输出如下

```
  File: '/usr/bin/wc'
  Size: 48832           Blocks: 96          IO Block: 4096   regular file
Device: fd00h/64768d  Inode: 1522           Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Context: system_u:object_r:bin_t:s0
Access: 2017-03-07 22:32:33.811018027 +0800
Modify: 2016-11-01 01:21:49.000000000 +0800
Change: 2017-03-04 11:57:02.192098765 +0800
```

它几乎包括了 Linux 文件的全部属性。其中，“File”为文件名；“Size”为文件大小；“Blocks”为文件所占 512 字节的块数；“IO Block”为文件系统 I/O 块大小；“regular file”表明这是一个普通文件（“block special file”：块设备文件；“character special file”：字符设备文件；“fifo”：命令管道设备文件；“directory”：目录文件）；“Device”文件所在设备 ID；“Inode”

为 I 节点号；“Links”为硬链接数；“Access”为文件访问权限；“Uid”为用户标识；“Gid”为组标识；“Context”为 SELinux 上下文；“Access” 为访问时间；“Modify” 为修改时间；“Change”为属性修改时间。

3．修改文件存取时间或创建空文件（touch）

1）功能及用法

touch 命令的功能有两点：一是建立空文件；二是更新文件的存取时间。默认时，将文件属性中的 3 个时间都修改为系统的当前时间，若指定文件不存在，则创建空文件。touch 命令的用法为：

```
touch   [-acm] [-r ref_file] [-t [[CC]YYMMDDhhmm[.ss]] files
touch [-acm]   [-t MMDD[yy]]   files
```

2）参数说明

touch 命令的部分选项如表 3-17 所示。

表 3-17　touch 命令的部分选项

选　　项	功 能 描 述
-a	改变文件的访问时间
-m	改变文件的修改时间
-c	假如目的文件不存在，禁止创建新文件
-d datestr	设定日期。datestr 支持自然语言日期和时期格式，必要时使用引号括起来
-t timestamp	设定文件日期和时间日戳。timestamp 格式为[[CC]YY]MMDDhhmm[.ss]

3）示例

```
#touch myfile                              #将文件的访问时间等改为当前时间，若不存在则创建之
#touch -a -t 200601010030 myfile yourfile  #修改文件的存取时间
#touch -m -t 200706050403 myfile           #修改文件的修改时间
#touch -m -d "Feb 2 2006 18:20" /tmp/tmpf
```

3.2.4　文本文件编辑与操作基本命令

1．文本编辑命令（vi）

1）功能及用法

vi 是 UNIX/Linux 系统基本文本文件的全屏编辑器。

vi 一次可以编辑多个文件，若 vi 启动时没有指定文件名，则 vi 将创建一个无名的工作文件，待用户保存时由用户指定文件名。若用户指定的文件不存在，则 vi 就创建一个新文件。若用户对文件的修改不保存，则原文件不发生作用。

设 vi 编辑的文件为 file，则 vi 在工作时，在与被编辑文件相同的目录内创建一个名为.file.swp 的临时文件，若没有指定文件名，则在当前目录下创建.swp 临时文件，此临时文件也叫交换文件。当 vi 结束对当前文件编辑后，对应的临时文件被自动删除，但当 vi 非正

常退出时，此临时文件就会被残留下来，在下次再编辑此文件时会出现错误信息。Vi 的整个工作过程和被编辑的文件无关，只是在保存的时候才修改被编辑的文件。

vi 有三种工作模式：命令模式、编辑模式和命令项模式，各模式的功能和区别如下。

（1）命令模式：当进入 vi 时，它处在命令模式。在这种模式下，用户可通过 vi 的命令对文件内容进行处理，如删除、移动、复制等，也可通过插入命令进入编辑模式。

（2）编辑模式：用户可在命令模式下通过 o，O，a，A，i，I 等命令使 vi 进入编辑模式。在编辑模式下，用户能在光标处输入内容，或通过光标键移动光标，也可通过按 Esc 键返回命令模式。

（3）命令项模式或叫底行模式：在命令项模式下用户输入冒号“:”后，光标会跳到底行，输入相关命令后可完成指定操作。

vi 的使用方法为：

```
vi [ options ]   [ files ]
```

2）参数说明

vi 的部分参数如表 3-18 所示。

表 3-18　vi 命令的部分参数

选　　项	功 能 描 述
[file]	从 file 第一行开始。当不提供文件名时，打开一个无名文件
+[#] file	进入 vi 后光标直接定位到第#行首，若不指定#，则光标定位到末行首
+/pattern file	从指定的模式匹配行行首处开始
-r/-L	列出当前目录内所有因 vi 非正常退出而残留的交换文件，以便恢复
-r <.*.swp>	从交换文件.*.swp 中恢复文件
-r file	从 vi 崩溃或非正常退出中恢复文件
-R	以只读方式编辑文件。使用此参数时，被编辑的文件不能回存，但可以另存

3）vi 的基本操作

（1）编辑模式。在命令模式下，用户可使用表 3-19 所示的命令进入编辑模式，编辑完毕后通过按 Esc 键返回命令模式。

表 3-19　vi 的插入功能键

键 或 命 令	功 能 描 述	键 或 命 令	功 能 描 述
i	光标处插入	I	行首插入
a	光标后追加	A	行末追加
o	当前行后插入新行	O	当前行前插入新行

（2）删除命令。在命令模式下可以通过表 3-20 所示的命令删除文字。

表 3-20　vi 的删除命令

命　　令	功 能 描 述	命　　令	功 能 描 述
x	删除光标处的字符	X	删除光标前的字符
d0	从光标处删除到行首	D	从光标处删除到行末

（续表）

命　令	功 能 描 述	命　令	功 能 描 述
db	删除光标到本 word 始	dw	删除光标到下一个词首
dd	删除光标所在的行	#cmd	对命令 cmd 执行#次重复

#cmd 是对命令状态可执行命令的重复执行，cmd 是命令，#为重复次数。例如，3x，将从光标处删除 3 个字符，相当于执行 3 次 x 命令。5dw 将删除光标后的 5 个 word 或重复执行 5 次 dw。

（3）修改或替换。在命令模式下输入表 3-21 所示命令可修改或替换文字。

表 3-21　vi 的修改与替换命令

命　令	功 能 描 述	命　令	功 能 描 述
cc/S	替换光标所在行	s	替换光标处的字符
C/R	修改当前行光标后的部分	r	替换当前一个字符
cw	修改光标到一个字的结尾部分	cb	修改光标到一个字的开始部分

（4）搜索与定位。在命令模式下可使用表 3-22 所示命令进行字符串搜索或定位光标。

表 3-22　vi 的搜索与定位命令

命　令	功 能 描 述	命　令	功 能 描 述
0	定位至行首	w	光标移至本行下一字开始
$	定位至行末	b	光标移至本行前/当前字开始
:#|#G	定位到第#行首	E	光标移至本行下一/当前字尾
:0|gg	定位到首行行首	%	查找单括号对（）
:$|G	定位到末行行首	[[|]]	定位到最外层{}的{/}
#%	光标移至#%处行首	[{|]}	定位到当前层{}的{/}
/exp	后向定位到字符串 exp	[/|]/	定位到注释块的开始/结束
?exp	前向定位到字符串 exp	Fc|fc	后/前向定位到当前行字符 c
/^exp	定位到行首为 exp 的行	Tc|tc	后/前向定位到当前行字符 c 后/前
/exp$	定位到行末为 exp 的行	g0|gm|gl	光标移动到行首/中/末
n,m?|/exp	在 n-m 行前/后向搜索 exp	gj|gk	光标下/上移一屏幕行（而非文本行）
n | N	前/后向继续查找	*|#	前/后向搜索光标下的单词

（5）搜索和替换。在命令项模式下输入如表 3-23 所示命令可进行文字搜索与替换。

表 3-23　vi 的搜索与替换命令

命　令	功 能 描 述
:s/exp1/exp2/g	将当前行的 exp1 替换成 exp2
:n,ms/exp1/exp2/g	将从 n 到 m 行中的 exp1 替换为 exp2
:n,mg/exp1/s//exp2/g	将从 n 到 m 行中的 exp1 替换为 exp2
:g/exp1/s//exp2/g	将全文中的 exp1 替换为 exp2

说明：在表示行和行范围时，还有以下规定："."表示当前行；%表示"1,$"，即从第一行到最后一行的整个文档；"1,."表示从第 1 行到当前行；".,$"表示从当前行到最后一行。

（6）块操作命令。在命令项模式下输入表 3-24 所示命令可进行文件内容的块移动、复制、删除和保存等操作。

表 3-24　vi 的块操作命令

命　　令	功 能 描 述	命　　令	功 能 描 述
:n1,n2d	删除 n1 到 n2 行的内容	:n1,n2tn3	将 n1 到 n2 行内容复制到 n3
:n1,n2mn3	将 n1 到 n2 行内容移动到 n3	:n1,n2w[!] [file]	将 n1 到 n2 行内容写入[file]

（7）缓冲区操作命令。vi 默认时使用无名缓冲区进行工作，比如，将文件的某些部分删除了，但可以用 u 命令进行撤销。文件内容的复制与移动都是通过缓冲区进行的。vi 还可使用命名缓冲区 a-z，然后通过缓冲区进行复制或粘贴等操作。用法如表 3-25 所示。

表 3-25　vi 的缓冲区操作命令

命　　令	功 能 描 述
Y/yy	将当前行放入无名缓冲区
P/p	将无名缓冲区中的内容粘贴到当前行的上/下方
"cyy	将当前行放入缓冲区 c（c 为任何小写字母）
"cd	将当前行删除，内容放入缓冲区 c
"cP/"cp	将缓冲区 c 中的内容粘贴到当前行的上/下方

说明：对任何使用各种删除命令删除的内容都被保留在无名缓冲区中，用户可以使用 p 命令在光标后追加它们。因此，命令串 xp 可以交换光标处及其后两个字符的位置；命令串 ddp 可以交换光标所在行及其下一行的位置。

（8）其他命令。在使用 vi 命令时，还可以使用一些命令进行行号设置、字符大小写转换等工作，相关命令如表 3-26 所示。

表 3-26　vi 的其他命令

命　　令	功 能 描 述	命　　令	功 能 描 述
.	重复上次执行的命令	u	作废刚刚执行过的命令
:set nu	设置行号	:set nonu	取消行号
J	将下一行合并到当前行	~	将光标处的字符进行大小写转换
Ctrl+G	显示当前编辑文件信息	Ctrl+L	刷新屏幕（屏幕乱时使用）
<<	把当前行向左移动一个 tab 位	>>	把当前行向右移动一个 tab 位
:n	定位到下一文件	:N	定位到上一文件
:r file	读入文件到光标处	:!cmd	执行 shell 命令 cmd
:!!	重复上次执行的 shell 命令	:r!cmd	执行 cmd 并将其输出追加到光标处
:e!	重新读入当前文件，作废以前所有操作	:e file	放弃当前文件编辑，以 file 启动 vi

（9）退出。当用户对文件编辑完成后，可以采用如表 3-27 所示的命令保存文件或退出 vi。

表 3-27　vi 的退出

命　　令	功 能 描 述	命　　令	功 能 描 述
:w	保存文件，但不退出	:wq[!]	保存退出
:x	保存文件，且退出 vi	:w>>file	将现行文件的内容追加到文件 file
:q	退出（若文件已更改，则不能退出）	:w file	将现行文件的内容写入 file
:q!	不保存退出	:w! file	将现行文件内容写入已存在的 file
ZZ	保存文件且退出 vi	ZQ	不保存退出

说明：

（1）在 Linux 系统中，vi 还有很多其他命令，可谓是功能强大，这里只介绍它的基本部分。

（2）在 Linux 系统中还有很多用于文本编辑的其他命令，如 ex、view、vim、emacs 等，另外还有用于图形界面的编辑器，如 gedit 等，用户可以有选择地使用它们。

2．字符串过滤命令（grep）

1）功能与用法

grep 的功能是字符串搜索与过滤。用于字符过滤的命令有 3 个，分别是 grep、egrep 和 fgrep，三者都用于在指定文件或流中搜索指定字符串，并显示出它所在行的相关信息。默认情况下，grep 使用标准正则表达式，egrep 使用扩展正则表达式，fgrep 则使用固定的字符串。以 grep 为例，它的用法为：

```
grep [options] pattern [files]
grep [options] [-e pattern | -f patternfile] [files]
```

若不指定搜索的文件，则默认为标准输入。

2）参数及说明

grep 命令的部分参数如表 3-28 所示。

表 3-28　grep 命令的部分参数

选　　项	功 能 描 述
-A/-B/-C num	在匹配模式的行后、行前、前后各输出 num 行内容
-a,--text	将二进制文件作为文本文件来对待
-b,--byte-offset	在每行前显示以字符为单位的偏移量
-c,--count	不显示匹配内容，只显示匹配行数
-D read/skip	对设备文件可以采取的动作：读或跳过
-d read/skip/recurse	对目录文件可以采取的动作：读、跳过或递归
-E/-F	使用扩展正则/固定表达式，调用 egrep/fgrep
-e pattern	使用表达式 pattern，用于保护以“-”开头的表达式
-f patternfile	从 patternfile 中读取表达式
-h,--no-file-name	输出结果前面不带有文件名，默认时每行前均有该行所在的文件名
-I	忽略二进制文件

（续表）

选　　项	功 能 描 述
-i,--ignore-case	忽略大小写的区别
-l	若有匹配，每个文件只输出一个文件名，每个文件搜索遇到匹配时立即退出
-m NUM	在文件中搜索到 NUM 次出现时停止本文件搜索
-n,--line-number	输出时显示行号
-o	只显示行中的匹配部分，而非整行
-q,--quiet	无输出，只有返回码。可用$?进行访问
-s,--no-messages	不显示工作过程中的错误信息
-U，--binary	文件按二进制对待
-v,--invert-match	反向搜索，显示不匹配的内容
-w/-x	整字/行匹配模式
-Z，--null	输出空字符(\0)代替回车字符

3）grep 搜索示例

（1）使用扩展正则表达式在 file 文件内搜索 someone、Someone、Anyone 或 anyone：

```
#grep -E '([Ss]ome)|([Aa]ny)one' file
```

（2）使用扩展正则表达式在 file 文件内搜索 henry、Henry、henrietta 或 Henrietta：

```
#grep -E '[Hh]enr(y)|(ietta)' file
```

（3）使用固定表达式在/etc/passwd 内搜索 root 用户：

```
#grep -F root /etc/passwd
```

（4）多文搜索。使用一般正则表达式在 a*、b*和 c*中搜索 mytext：

```
#grep mytext [abc]*
```

（5）流搜索。在目录/dev 内搜索文件名中含字符串 sda 者：

```
#ls /dev | grep 'sda'
```

（6）反向搜索，在文件/etc/inittab 中搜索非注释行（行首字符不为#的行）：

```
#grep -v "^#" /etc/inittab
```

3．文件排序命令（sort）

1）功能与用法

sort 命令对指定文件中的行按要求进行排序，并将结果写到标准输出。如果指定多个文件，则 sort 命令将这些文件连接起来，并当作一个文件进行排序。如果不指定文件名或文件为-，则使用标准输入。如果不指定排序要求，则 sort 以行为单位按 ASCII 码的顺序排序。sort 命令的用法为：

```
sort [options] [files]
```

2）参数说明

sort 命令的部分参数如表 3-29 所示。

表 3-29　sort 命令的部分参数

选　　项	功 能 描 述
-b	忽略前导空格
-c	检查给定文件是否已排序，如果没有都排序，则打印错误信息后退出，并返回 1
-d	忽略特殊字符。按字典顺序排序，仅字母、数字、空格和制表符有意义
-f	忽略大小写
-i	忽略不可打印字符
-k POS1[,POS2]	以从位置 POS1 开始（到 POS2 为止）的字符为“键”排序。若指定 POS2，则为从 POS1 到 POS2，否则为 POS1 开始到行末。POS 结构为 F[.C][OPTS]，F 为域编号，C 为域内字符位置，OPTS 为一个或多个字符
+posl -pos2	以相对数定义列位置
-m	若给定文件已排好序，则合并文件
-n	按字符串数值排序
-o FILE	指定输出文件。可与输入文件名相同
-r	按逆序输出排序结果
-t SEP	指定字段分隔符为 SEP，默认为空格
-u	除去重复的行，使文件中的每一行唯一

3）示例

（1）对文件按字典顺序排序：

```
#sort filei > fileo                #将 filei 按字典顺序排序后输出到 fileo
```

（2）以第二个域为键进行排序：

```
#sort -k 2,2 filei                 #使用 k 指定绝对域号
#sort +1 -2 filei                  #使用相对量定义域号
```

（3）以第二个域的第一个字符为键进行排序：

```
#sort -k 2, 2.1 ifile
#sort +1.0 -2.1 ifile
```

说明：本地环境变量的设置将影响 sort 的排序结果。

4．处理文件中的重复行命令（uniq）

1）功能及用法

uniq 命令用于处理文件中的重复行。该命令首先比较相邻的行，然后处理（删除或显示）其后重复的行，在文件中不相邻重复行的将不会被处理。若要想发现某文件中的所有重复行，可先用 sort 命令进行排序。uniq 命令将处理的结果输出到标准输出或指定文件，一般不会对输入文件造成影响，除非指定的输出文件与输入文件名相同。uniq 命令的用法为：

uniq [options] [infile [outfile]]

2）参数说明

uniq 命令的部分参数如表 3-30 所示。

表 3-30 uniq 命令的部分参数

选 项	功 能 描 述	选 项	功 能 描 述
-c	在输出行前加入在输入文件中连续出现的次数	-s *N*	忽略由 *N* 指定数目的字符
-d	仅显示重复行	-[f] *N*	忽略由 *N* 变量指定数目的字段
-D	显示所有重复行	-w *N*	只比较前 N 个字符
-u	仅显示不重复的行	-i	比较时忽略大小写

3）示例

```
#uniq ifile ofile              #将输入文件中的非重复行保存到另一个文件中
#uniq ifile > ofile            #同上
#uniq –d ifile                 #仅显示重复行
#uniq –D ifile                 #显示所有重复行
```

说明：默认情况下，uniq 以整行方式比较输入文件的所有行；输入和输出文件不能相同，否则不能保证得到正确结果；本地环境决定了-f 标志使用的域分隔符及-s 标志如何将字节解释成字符。

5．文件内容信息统计（wc）

1）功能及用法

wc 命令的功能是对输入文件的信息进行统计。统计信息包括行数、单词数和总字节数等。若有多个输入文件，则将对每个输入文件进行分别统计，在最后还给出所有文件的统计信息。wc 的用法为：

wc [-c] [-m] [-w] [-l] [-L] files

2）参数说明

wc 命令的部分参数如表 3-31 所示。

表 3-31 wc 命令的部分参数

选 项	功 能 描 述	选 项	功 能 描 述
-c/-m	字节/字符数统计	-w	统计单词数
-l	统计行数	-L	统计最长行的字节数

3）示例

```
#wc /etc/inittab               #综合统计/etc/inittab 的内容信息
#wc -w /etc/inittab            #统计/etc/inittab 的单词数
#wc -l /etc/inittab            #统计/etc/inittab 的行数
#wc -c /etc/inittab            #统计/etc/inittab 的字符数
```

3.2.5　日期、时间与时区管理命令

UNIX/Linux 提供有 date、hwclock、timedatectl 等命令用于日期和时间的管理，除此之外，桌面或工作站系统还可在图形界面下对日期和时间进行管理。第 13 章还要介绍通过 NTP 实现系统时间的自动校正。

1．时区及相关文件

时区是日期和时间的基础，必须设置正确，否则系统时间与手表的时间就可能不一致。对于有营业时间限制的业务系统更是如此，否则可能造成正在营业时间但系统却不工作了，或者非营业时间系统却还正在工作。前者可以造成客户不满，后者可以产生安全隐患。因此，系统时间不能随意改变。

系统的时区控制文件为/etc/localtime，它是/usr/share/zoninfo/内某个文件的复制品或链接，在中国，这个文件是/usr/share/zoninfo/Asia/Shanghai。因此，设置中国时区的方法之一是执行命令：

```
#ln -f -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
```

日期和时间的设置与显示受时区环境变量 TZ 的影响，在 Linux 下，格林尼治标准时区是 TZ=UTC，中国的标准时区是 TZ=CST。

2．日期与时间管理命令（date）

1）功能与用法

date 命令的用法为：

```
date [options] [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

2）参数说明

date 命令的部分参数如表 3-32 所示。

表 3-32　date 命令的部分参数

选　　项	功 能 描 述
MMDDhhmm[CC]YY][.ss]	设置系统的时间格式
+form	指定时间显示格式
-d datestr	显示 datestr 中所设定的时间（非系统时间）
-s datestr	将系统时间设为 datestr 中所设定的时间
-u	显示当前的格林尼治时间

在显示方面，用户可以指定显示的格式，格式为+form，一个“+”后接数个格式显示控制标记（参见表 3-33 和表 3-34）。如果是不以“+”作为开头，则表示设定时间，而时间格式为 MMDD hhmm[[CC]YY][.ss]，其中 MM 为月份，DD 为日，hh 为小时，mm 为分钟，CC 为世纪，YY 为年份后两位数字，ss 为秒数。只有 root 用户才能设置时间。

表 3-33　date 显示格式 1（日期显示控制格式）

标　　志	作　　用	标　　志	作　　用
%n	回车	%t	Tab 符
%a	星期几（英语简拼，如 Sun）	%A	星期几（英语全拼，如 Sunday）
%b/%h	月份（英语简拼，如 Jan）	%B	月份（英语全拼，如 January）
%c	时间和时间（Thu Mar 3 23:05:25 2005）	%C	Century，年的高 2 位，比如 20
%d	月日期（不足 2 位者前补 0）	%D	MM/DD/YY 格式日期，同%m/%d/%y
%e	月日期（不足 2 位者前补空格），同%_d	%F	YYYY-MM-DD 格式日期，同%Y-%m-%d
%g/%y	2 位年份	%G/%Y	4 位的年份
%j	年中天数	%m	2 位月份（不足 2 位者前补 0）
%u	周日期（1～7,1 对应星期一）	%U/%V/%W	年中周数（00~53）（三者有区别）
%w	周日期（0～6,0 对应星期天）	%x	MM/DD/YYYY 格式日期

表 3-34　date 显示格式 2（时间显示控制格式）

标　　志	作　　用	标　　志	作　　用
%H	时（00～23）（2 位，不足者前补 0）	%I	时（01～12）（2 位，不足者前补 0）
%k	时（0～23）（2 位，不足者前补空格）	%l	时（1～12）（2 位，不足者前补空格）
%M	分（00～59）（2 位，不足者前补 0）	%N	纳秒（000000000~999999999）
%p	AM 或 PM	%P	am 或 pm
%r	12 小时制时间，如 11:10:09 PM	%R	24 小时制时间，同%H:%M
%s	从 1/1/1970 00:00:00 UTC 到目前的秒数	%S	时间秒（00~60）
%T	24 小时制时间，同%H:%M:%S	%X	本地时间，如 07:58:35 PM
%z	数字格式时区，可用的还有%:z,%::z	%Z	符号时区，如 CST

3）日期和时间使用示例

```
#date                              #以默认格式显示系统的当前时间
#date +%T%n%D                      #按指定格式显示系统当前时间，时间和日期各占一行
#date 05311010                     #设置系统时间为当年 5 月 31 日 10:10:00
#date +"%Z%t%z%t%:z%t%::z%:::z"    #显示不同格式的时区
#date -s "+2 minutes"              #将系统时间前提 2 分
##将从格林尼治标准时间 1970-01-01 0:0:0 到时间字符 d 的时间转换为秒
#date -d "2017-05-01 00:00:00 CST" +%s        #仅转换而不设置时间
#date -d "2017-05-01 00:00:00 GMT" +%s        #同上
```

3．hwclock

1）功能及用法

hwclock 的功能是访问硬件时钟。它可以完成以下功能：显示硬件时钟；将硬件时钟设置为指定时间；将硬件时间设置为系统时间，或相反地将系统时间设置为硬件时钟；比较系统时间与硬件时间；补偿硬件时钟漂移等。其用法为：

```
hwclock [function] [options]
```

2）子功能及参数

hwclock 的子功能（function）如表 3-35 所示。

表 3-35　hwclock 的常用功能

功　　能	描　　述
-c | --compare	周期性地（每 10 秒）比较系统和硬件时钟（^C 结束）
-r | --show	读取硬件时间并以 ISO 8601（yyyy-mm-dd hh:mm:ss.* +TZ）格式显示
-s |--hctosys	由硬件时钟设置系统时间
--set | --date=date_string	由--date=date_string 格式的时间设置硬件时钟，同时更新/etc/adjtime 中的时间戳
-w | --systohc	由系统时间更新硬件时间，同时更新/etc/adjtime 中的时间戳

hwclock 的常用选项有——localtime，-u/--utc 用于告诉相关子功能按本地时间或格林尼治标准时间处理。

3）示例

```
hwclock --show                                   #读取硬件时间并以 ISO 8601 格式显示
hwclock --set --date "dd mmm yyyy HH:MM"         #设置硬件时间
hwclock --set --date "yyyy-mm-dd HH:MM" --localtime  #将本地系统时间设为硬件时间
hwclock --systohc --localtime                    #将本地系统时间同步为硬件时间
hwclock --hctosys                                #将硬件时间同步为系统时间
```

4．timedatectl

1）功能及用法

timedatectl 的功能是设置和显示系统时间。其用法为：

```
timedatectl [options] {cmd}
```

2）子功能及选项

cmd 是 timedatectl 的子功能，常用的子功能如表 3-36 所示。

表 3-36　timedatectl 的子功能

子　功　能	描　　述
list-timezones	列所有可用时区
status	显示当前与时间相关的配置信息
set-time TIME	设置系统和硬件时间，时间格式为“YYYY-MM-DD hh:mm:ss”。等价于 date -s && hwclock -w
set-timezone TIMEZONE	设置时区（中国时间为 Asia/Shanghai）
set-local-rtc BOOL	通常系统使用 UTC，可用此设置是否使用 UTC。BOOL：0/no—UTC，1/yes—本地时区
set-ntp BOOL	设置是否启用网络时间。BOOL：0/no—不使用网络时间；1/yes—使用网络时间

与 set-local-rtc 可配合使用的还有一个选项--adjust-system-clock，让在 set-local-rtc 时也同步系统时间。

3）示例

```
#timedatectl status                                    #显示配置
#timedatectl set-time 12:30:00                         #设置时间
#timedatectl set-time 2017-03-31                       #设置日期
#timedatectl set-time "2017-03-31 12:30:00"            #同时设置日期和时间
#timedatectl list-timezones                            #显示所有时区
#timedatectl set-timezone Asia/Shanghai                #将时区设为 Asia/Shanghai
#timedatectl set-ntp yes                               #配置使用网络时间
```

说明：如果时间服务正在运行，则设置时间和日期可能失败，可能的错误信息为“Failed to set time: NTP unit is active”。

5．日历显示命令（cal）

cal 命令用来显示日历。其用法为：

```
cal [-smjy13] [[month] year]
```

其部分可用选项如表 3-37 所示。

表 3-37　cal 命令的部分可用选项

选　　项	功能描述	选　　项	功能描述
-1	只显示 1 个月日历（默认）	-m	以星期一为每周的第一天方式显示
-3	显示以当前月为中心的 3 个月日历	-j	按儒略历（Julian Calendar）显示日历
-s	以星期日为每周的第一天方式显示（默认）	-y	显示当年历

如果只有一个参数，那么它代表年份（1—9999），年份位数要写全。例如，cal 17 显示的不是 2017 年，而是公元 17 年的日历。

6．图形界面

对于桌面或工作站系统，可以通过“设置中心（Settings）”，或在 CLI 界面下执行 gnome-control-center 后选择“Date & Time”进入 Date 和 Time 管理界面，也可以在安装 system-config-date 软件包后，从 CLI 界面执行命令 system-config-date 进入 Date 和 Time 管理界面进行日期和时间设置。

3.2.6　进程管理基本命令

Linux 系统在启动时创建了很多进程，用户程序的运行也是通过进程方式实现的。进程的创建者称为父进程，被创建者称为子进程，子进程还可以创建子进程。父进程也好，子进程也好，在操作系统中都表现为进程。为了对进程进行管理，系统为每个进程分配了一个编号——进程号，进程号也叫进程标识，用 pid（process identification）来表示。在诸多的进程、父进程、子进程之间按先后创建关系构成了一棵进程树，进程树上的每一个分支都对应或描述着一组进程的家族关系。进程树可通过命令 pstree 来查询。

进程之间可以通过信号进行通信，以实现进程间一定程序的控制，用于进程间信号通信

的命令是 kill 和 trap。

1．信号及默认处理方式

UNIX/Linux 中使用很多信号（参见表 3-38），用于进程通信与控制。

表 3-38　Linux 系统的部分信号

信号名称	信号数值	默认动作	意义和功能描述
SIGHUP	1	终止运行	挂机或断线信号，可能是终端控制程序死亡
SIGINT	2	终止运行	收到键盘 Ctrl_C（有的系统为 Delete）信号
SIGQUIT	3	写出进程映像	收到键盘退出信号（Ctrl_\）
SIGILL	4	同上	非法指令
SIGTRAP	5	同上	单步执行，程序跟踪
SIGABRT	6	同上	ABORT 信号
SIGBUS	7	同上	总线错误（非法内容访问）
SIGFPE	8	同上	协处理器错误
SIGKILL	9	终止运行	KILL 信号，不可屏蔽
SIGUSR1	10	同上	用户自定义信号 1
SIGSEGV	11	写出进程映像	段错误
SIGUSR2	12	终止运行	用户自定义信号 2
SIGPIPE	13	同上	断的管道：有写者无读者
SIGALRM	14	同上	定时器信号
SIGTERM	15	同上	进程终止信号
SIGSTKFLT	16	同上	协处理器堆栈错误
SIGCHLD	17	忽略	子进程退出
SIGCONT	18	继续执行	继续信号。暂停者一旦收到此信号，便继续运行
SIGSTOP	19	停止执行	停止执行信号 STOP，不可屏蔽
SIGTSTP	20	停止执行	键盘输入的 stop 信号（Ctrl_Z）
SIGTTIN	21	停止执行	后台进程读 tty
SIGTTOU	22	停止执行	后台进程写 tty
SIGURG	23	忽略	Socket 上的紧急事件（带外数据可用）
SIGXCPU	24	写出进程映像	超出 CPU 时限
SIGXFSZ	25	写出进程映像	超出文件最大限量
SIGVTALRM	26	终止	虚拟定时器报警
SIGPROF	27	终止	统计信息定时器报警
SIGWINCH	28	忽略	窗口大小变化
SIGIO	29	终止	异步 I/O 事件
SIGPWR	30	停止执行	电源失效

在 UNIX/Linux 的众多信号中，有的是硬件产生的，有的是软件产生的，还有一些是人工可以制造的。这些人工可以制造的信号简称为人工信号，它们是 SIGHUP、SIGINT、

SIGQUIT、SIGKILL、SIGTERM、SIGUSR1 和 SIGUSR2 等，或简写为 HUP、INT、QUIT、KILL、TERM、USR1 和 USR2 等。这些人工信号中的一些是可以被捕获的，捕获命令为 trap。

2．进程树及进程状态查询（pstree）

1）功能及用法

pstree 命令用于显示系统内进程间的关系——进程树。将系统中进程间的关系以树状图显示，系统中所有进程的进程树都会以基本进程 init(1)为根，如果指定用户，则只显示此用户拥有的进程子树。pstree 的用法为：

```
pstree [-a] [-c] [-h|-Hpid] [-l] [-n] [-p] [-u] [-G|-U] [pid|user]
```

2）参数说明

pstree 命令的部分参数如表 3-39 所示。

表 3-39　pstree 命令的部分参数

选　　项	功 能 描 述
pid	显示指定进程 pid 的子进程树
user	显示指定用户 user 的子进程树
-a	显示完整的命令行参数，包含 pid，参数或内存驻留情况（括号内的表示被换出）
-c	不使用精简方式
-h	突出显示正在执行的进程
-H<pid>	突出显示指定进程，若指定进程不存在，则 pstree 执行失败
-l	长列格式显示树状图，默认情况下超长者将被截去
-n	按 pid 排序，而非默认进程名排序
-p	显示 pid
-u	显示用户名称

3）示例

```
#pstree                        #显示系统进程树
#pstree -p | grep vsftpd       #查询进程 vsftpd 的进程信息
#pstree user                   #显示指定用户（user）的进程
```

3．终止进程执行（kill）

1）功能及用法

kill 命令的功能是向指定进程（pid）发送一个特定的信号（signal）。默认情况下，kill 发送终止信号 TERM，其值为 15。按照系统规定，当进程收到某信号后，应该采取相应动作，默认动作是结束执行。kill 的用法为：

```
kill [-s signame | -n signum | -signum] pid ...
kill -l [signame]
```

2）参数说明

kill 命令的部分参数如表 3-40 所示。

表 3-40　kill 命令的部分参数

选　　项	功 能 描 述
pid	进程编号
-l [signame/signum]	列出所有可用的信号或指定信号的名称/值
-s signame -n signum -signum	发送的信号，可用的信号有 HUP(1)、INT(2)、QUIT(3)、KILL(9)和 TERM(15)等，另外还有系统为用户保留的 USR1 和 USR2

3）使用举例

（1）列出可用信号。

```
#kill -l                        #列出所有信号（参见表 3-38）
```

（2）终止已知 PID 的进程。

```
#kill -9 3231                   #发送信号 9（KILL）终止 pid 为 3231 的进程
```

（3）终止已知进程名的进程。

```
##终止名为 named 的进程
#pstree –p | grep named         #查询进程信息
  |-named(1656)                 #命令输出，由此可见 named 的 PID 为 1656
##可使用以下命令中的任一个终止 PID=1656 的进程 named
#kill -9 1656                   #向 pid=1656 的进程发信号 9，终止它
#kill -KILL 1656                #同上
#kill -SIGKILL 1656             #同上
```

4．信号的捕获（trap）

1）功能及用法

我们通常只处理一些人工软中断信号，在这些信号中 SIGHUP、SIGINT、SIGQUIT、SIGUSR1、SIGUSR2 和 SIGTERM 是可屏蔽的，而 SIGKILL 是不可屏蔽的。

通常，对一个信号的处理方法有 3 种：

（1）系统默认（一般是终止当前进程的执行）;

（2）忽略;

（3）定义一个新动作。通过命令 trap 可以完成这些工作，trap 用法为：

```
trap [-lp] [ [arg] signals]
```

2）参数说明

选项-l 用于列出可用信号（trap -l 等于 kill -l）；-p 用于显示所有或指定信号已经定义的动作；signals 为信号；arg 为命令或由“;”连接的命令序列或 shell 函数（见后述），表示当信号 signals 到来时将执行由 arg 规定的动作。

① 如果 arg 被忽略（缺或不存在），则 signals 的所有行为将被自动恢复为调用 trap 前的设定。

② 如 arg 是空串，则 signals 将被忽略。

③ 如果 trap 没有选项和参数，则显示当前所有已定义的信号动作。

3）信号捕获示例

```
#trap 'echo -e "You pressed ^C!\n"' 2       #捕获 2，并指定处理方法
#trap -p 2                                  #显示 INT 信号动作
##按一下组合键^C，观察一下效果
#trap                                       #列出当前信号动作
#trap '' 2 3 15                             #捕获并忽略它们。''为两个单引号，中间无内容
##再按组合键^C，观察一下效果
#trap 2 3 15                                #恢复信号 2，3，15 的之前动作
##再按组合键^C，观察一下效果
```

5．进程挂起或作业的前/后运行切换（fg、bg 和 jobs）

当一个交互式进程正在运行时，用户可以通过按^Z 将它挂起。之后，可让其在后台等待运行，也可让其从后台再重新回到前台运行。实现这些功能的命令是 fg 和 bg，它们都是 bash 的内部命令，还有 jobs 命令用于这些作业队列查询和管理。它们的用法为：

```
fg [job]
bg [job]
jobs [options] [job …]
```

fg（foreground）的功能是让被挂起的进程或作业到前台来运行；bg（background）的功能是让被挂起的进程或作业在后台运行，就像用符号“&”启动的那样。若不指定 job 则默认使用当前作业。

若用户正在运行 man bash，此时可以通过 Ctrl_Z 将其挂起，屏幕显示以下信息后并出现提示符：

```
[1]+   stopped   man bash
```

表明名为 man 的进程被挂起，作业编号为 1，+表示此为当前作业，用户可以在提示符下继续做其他工作。如果再启动一个进程，如计算器 bc，此时进入 bc 交互界面，用户同样可以通过 Ctrl_Z 将其挂起

```
[2]+   stopped   bc
```

系统回到提示符。运行 jobs 命令可得以下输出：

```
[1]-   stopped   man bash
[2]+   stopped   bc
```

由此可见，有两个作业，编号分别为 1 和 2，其中，2 为当前作业。此时可以通过命令

```
#fg [job]             #或
#bg [job]
```

将作业调到前台或放在后台运行，job 为 jobs 命令列出的作业号，如 1，2 等。如果输入

```
#fg                   #或
#fg 2
```

将使 bc 调到前台执行，而输入

```
#fg 1
```

将使作业 man 调到前台执行。同样输入

```
#bg 1
```

也可使 man 命令在后台运行，但交互式命令在后台运行没有意义。

3.2.7　文件或目录比较命令

1．比较两个文件的内容（cmp）

1）功能与用法

cmp（compare）命令用于比较两个文件是否有差异。默认情况下，若发现有差异则标示出第一个不同的字符和行号后退出。cmp 命令的用法为：

```
cmp [ -l ] [ -s ] file1 file2 [ skip1 [ skip2 ]]
```

2）参数说明

cmp 命令的部分参数说明如表 3-41 所示。

表 3-41　cmp 命令的部分参数说明

选　　项	功 能 描 述
-l	显示出所有不同的位置和差异
-s	只返回退出码。0：文件相同；1：文件不同；>1：错误
skip1,skip2	分别表示 file1 和 file2 开始的偏移位置

3）示例

```
#cmp prog.o.bak prog.o            #确定两个文件是否相同
#cmp -l prog.o.bak prog.o         #显示两个文件的所有不同字节对
#cmp -s prog.c.bak prog.c         #比较两个文件，而不输出信息，可使用其返回值
```

说明：通常使用 cmp 命令比较非文本文件，使用 diff 命令比较文本文件。

2．比较文件的差异（diff）

1）功能与用法

diff（different）命令用于比较文本文件，且仅当输入为文本文件时才有效，它也用于比较目录。diff 命令的用法为：

```
diff [options] file1 file2
```

如果 file1 是目录，而 file2 不是，则 diff 将目录 file1 中与 file2 同名的文件与 file2 比较，反之亦然。当 file1 和 file2 都是目录时，diff 并不是将两个目录作为目录文件来比较的，而是比较其中相应文件的内容，若不使用-r 选项，则不做递归处理。

2）参数说明

diff 命令的部分参数如表 3-42 所示。

表 3-42　diff 命令的部分参数

选　　项	功 能 描 述	选　　项	功 能 描 述
-<#>	指定要显示#行的文本	-r	比较子目录中的文件
-a	强制 diff 比较二进制文件	-s	报告相同的文件，否则不提示
-b	忽略空格字符的不同	-t	在输出行扩展制表符
-B	不检查空白行	-T	在输出行前插入 Tab，默认为空格
-H	比较大文件时，可加快速度	-w	忽略所有空格
-F<file>	比较目录时，不比较文件名与 file 文件所包含 pattern 匹配的文件	-x<pattern>	比较目录时，不比较文件名与 pattern 匹配的文件
-l	输出时使用 pr 分页处理	-i	忽略大小写
-q	仅显示有无差异，不显示详细信息	-y	输出采用两文件并列输出方式

3）示例

```
#diff chap1.back chap1          #比较文件 chap1.bak 和 chap1 的不同处
#diff -w myp.c.bak myp.c        #比较两个文件，但是忽略空格字符个数的区别
#diff -r dir1 dir2              #比较目录 dir1，dir2
```

3．逐行比较两个文件已经排序的文件（comm）

1）功能与用法

comm（common）命令用于比较两个已排序文件的相同和不同之处，并将结果显示出来。comm 命令的用法为：

```
comm [-123] file1 file2
```

默认情况下，comm 输出包含 3 栏内容：第 1 栏为 file1 中的不同行，第 2 栏为 file2 中的不同行，第 3 栏为两个文件相同的行。

2）参数说明

comm 命令的参数说明如表 3-43 所示。

表 3-43　comm 命令的部分参数

选　　项	功 能 描 述	选　　项	功 能 描 述	选　　项	功 能 描 述
−1	禁止输出第一栏	−2	禁止输出第二栏	−3	禁止输出第三栏

3）示例

```
#comm -12 file1 file2           #显示两个文件中相同的行
#comm -23 file1 file2           #显示只出现在 file1 中的不同行
#comm -3  file1 file2           #只显示 file2 文件中不同的行
```

4．显示或截取文件行的指定部分（cut）

1）功能与用法

cut 命令用于从文件或标准输入中读取内容，并截取每一行的特定部分，然后将结果送到标准输出。cut 命令的用法为：

```
cut [ options ] [ files ]
```

2）参数说明

cut 命令的部分参数如表 3-44 所示。

表 3-44　cut 命令的部分参数

选　　项	功 能 描 述
-b List	指定字节列表
-c List	指定字符列表
-f List	指定文件中输出的域列表
-s	不输出不包含定界符的行
-d Sep	指定 Sep 作为输入文件域分隔符，默认为 Tab
--output-delimiter=Sep	指定 Sep 作为输出的域分隔符，默认无输出分隔符

在使用-b，-c 和-f 时，列表可用以下方式指定。N：第 N 个。N-M：从第 N 个到第 M 个。N-：从第 N 个到最后一个。-M：从第一个到第 M 个。N1-M1,⋯,Nn-Mn：逗号链接的多个连续列表。

3）示例

```
#cut -d: -f1 /etc/passwd                    #显示系统中的所有用户名
##显示以 Tab 分隔的组和组成员
#cut --output-delimiter=$'\t' -d: -f1,4 /etc/group
```

5．连接文件的行（paste）

1）功能与用法

paste 命令用于连接输入文件的行。默认情况下，paste 命令将每个文件的行当作一栏，并用制表符水平连接它们（并行合并）。顺序连接时，可以将 paste 命令作为 cat 命令。paste 命令的用法：

```
paste [ options ] [ files]
```

2）参数说明

-d Sep：指定在并行合并时使用 Sep 作为输出分隔符，默认为 Tab。-s：顺序合并文件。

3）示例

```
#paste file1 file2 file3                     #将多个文件按栏粘贴到一起
#paste -d "@" names places > emailusers      #用“@”作为栏分隔符
#ls | paste - - -                            #以 3 栏方式列当前目录
```

6．连接两个文件的行（join）

1）功能与用法

join 命令用于连接两个文件的数据字段。join 命令读取指定的文件，并根据“连续指标”连接文件中的行，将结果写到标准输出。这里的“连接指标”是指两个输入文件中具有相同值的（指定）域。被连接的文件必须为文本文件。join 命令的用法为：

join [options] file1 file2

2）参数说明

join 命令的部分参数如表 3-45 所示。

表 3-45　join 命令的参数

选　　项	功 能 描 述
-1 <filedid>	在第 1 个文件的 fieldid 域连接文件
-2 <filedid>	在第 2 个文件的 fieldid 域连接文件
-a #	除显示原输出外，还显示文件#中无相同“连接指标”的行。#为 1 或 2 代表文件 1 或 2
-e string	对于空的域使用 string 来填充
-i	比较域内容时，忽略大小写差异
-o<fmt>	按照指定格式输出结果。fmt=file:field, file:field,…
-t<c>	指定 c 为域分隔符
-v #	同-a，但仅显示文件#中无相同“连接指标”的行

3）示例

##在默认域（第 1 个文件的第 1 个域）上连接文件

#join -t: /etc/passwd /etc/group

##在默认域上连接文件，空域用字符串“---”代替

#join -e"---" -t":" /etc/passwd /etc/group

##连接文件并按指定格式输出（输出第 1 个文件的第 1,3,4 个域和第 2 个文件的第 4 个域）

#join -t: -e--- -o1.1,1.3,1.4,2.4 /etc/passwd /etc/group

7．文本文件排版（fmt）

1）功能与用法

fmt 命令的功能是从指定的文件里读取内容，将其按照指定格式重新编排后输出到标准输出，因此可用其进行简单“排版”。fmt 命令的用法：

fmt [-num] [options] [files]

2）参数说明

fmt 命令的部分参数如表 3-46 所示。

表 3-46　fmt 命令的参数

选　　项	功 能 描 述
-c	每段前缩排两列
-p string	仅合并含有指定字符串 string 前缀的行
-s	分隔长行，但不做格式处理
-t	首行与其他行有不同缩进

（续表）

选　　项	功能描述
-u	字间以一个空格间隔，句子间以两个空格分隔
-num	指定最大列宽，默认为 75（有的 UNIX 默认为 72）

8．文本文件包装（fold）

1）功能与用法

fold 命令的功能是将输入文件的长行分成多个固定宽度的行，然后将其输出到标准输出，默认行宽为 80 列。fold 命令的用法：

```
fold [ options ] [ files ]
```

2）参数说明

fold 命令的部分参数如表 3-47 所示。

表 3-47　fold 命令的部分参数

选　　项	功能描述	选　　项	功能描述
-b	按字节计算宽度，默认值是按列计数	-s	在空格处分行
-c	按字符计算宽度	-w #	以#的值为最大行宽，默认值为 80

3）使用示例

```
#fold -s -w 72 myfile            #将文件 myfile 按 72 列宽度，在空格处分行
```

3.2.8　其他操作命令

1．清屏命令（clear）

clear 命令用于清除用户的终端屏幕，使光标回到屏幕的左上角。其用法为：

```
clear
```

说明：在一般终端都可以使用组合键 Ctrl_L（^L）完成清屏操作。

2．字符串或变量输出命令（echo）

1）功能与用法

echo 命令将其命令行参数中的字符串或变量的值写到标准输出。默认情况下，每个 echo 命令执行完毕后会产生一个换行符，如果不带参数执行此命令，将输出一个空行。echo 命令的使用方式为：

```
echo [options] string ...
```

2）参数说明

echo 既是内部命令又是外部命令，内部命令的搜索一般先于外部命令。常用参数如下：-e 用于处理转义字符；-n 用于抑制默认的换行符；-E 用于抑制对转义字符的处理。

3）示例

```
#echo Please insert diskette …                          #显示字符串常量
#echo "My cell phone number is:\n13903750000"           #不处理转义字符
#echo [-e] "My cell phone number is:\n\t13903750000"    #处理转义字符
#echo -n "I am student, "                               #不处理换行
#echo "my name is Bill Joy"                             #以上命令的输出在同一行上
```

3．变量输入命令（read）

1）功能与用法

read 从标准输入上读入一行，并将它读到的内容按分隔符分隔的字符串传递给相应变量。若值的个数大于变量个数，则多余的部分赋给最后一个变量；若值的个数小于变量个数，则后面多余的变量被置空。如果没有指定变量名，则默认使用 REPLY 作为变量名。read 的用法为：

```
read [-d delim] [ -n num] [-p prompt] [-r] [-s] [-t time] var1 var2 …
```

2）参数说明

read 命令的部分参数如表 3-48 所示。

表 3-48　read 命令的部分参数

参　数	功　能
-d delim	指定新的分隔符
-n num	当 read 读到 num 个字符后返回
-p prompt	设置提示信息
-r	取消转义字符的转义作用。此种情况下不能使用继续行
-s	安静方式。键盘输入屏幕不回显，可用于密码输入
-t timeout	设置超时时间为 timeout。当等待时间超过 timeout 秒后返回

3）示例

```
#read x y                                    #输入：Hello World
#echo -e "x=$x\tVar y=$y"                    #x=Hello    y=World
#read -p "Name and Telephone number:" x y    #输入：Bill 123456
#echo "Name is $x, and Telephone is $y"
#read -s -p "Password Please: " p            #读取密码 p。输入时屏幕不回显
#echo "Password p=" $p                       #显示密码变量 p 的值
```

4．显示环境变量命令（env）

env（environment）命令用于显示当前环境变量或为一个将要运行的命令设置临时环境变量。如果不带参数运行，则它将显示用户所有当前环境变量。其用法为：

```
env [-|-i] [-u env_var] [var=value] … [cmd ]
```

参数-i 或-用于忽略所有继承的环境变量设置；-u env_var 用于取消 env_var 环境变量的设置；var=value 用于设置新的环境变量 var，其值为 value；cmd 为在新环境下执行的命令。

env 设置或取消的环境变量只对 cmd 的运行子环境有效，而不影响当前的环境变量。

env 的使用示例如下。

```
#date                          #显示当前系统日期
#env TZ=GMT date               #以新环境变量执行命令，显示格林尼治标准时间
```

5．环境变量的定义（export）

export 是 bash 的内部命令，用于环境变量的定义、修改和撤销。其用法为：

```
export [-fn] [name[=value]] ...
export -p
```

第 1 种用法用于定义、修改或撤销环境变量，参数-f 用于指定一个函数，-n 用于撤销一个环境变量。第 2 种用法用于显示当前 shell 通过 export 定义的环境变量。示例如下。

```
#my_env1="MyEnv1"              #定义本地或局部变量 my_env1
#export my_env1                #将 my_env1 变为环境变量
#export my_env2="MyEnv2"       #直接定义环境 my_env2
#env | grep my_env             #显示新定义环境变量定义
#echo $my_env1 $my_env2        #显示新定义环境变量的值
#export -n my_env1             #撤销对环境变量 my_env1 的定义
#env | grep my_env             #显示新环境变量的定义（只有 my_env2 了）
```

从示例可以看出，可以通过先定义变量，再执行 export 命令的办法定义环境变量：

```
my_env1="MyEnv1"; export my_env1
```

也可采用一步走的办法直接定义环境变量：

```
export my_env2="MyEnv2"
```

对于错定义或不再需要的环境变量可以重新定义或撤销。

6．定位可执行程序及相关信息（which、whereis、whatis 和 apropos）

find 命令可用于各种文件的查找与搜索（见后述），但 Linux 还给用户提供了 which、whereis、whatis 和 apropos 等专门用于对命令及其相关信息的搜索。

1）which

which 用于定位一个程序文件的位置，输出信息中包括别名和路径等。其用法为：

```
which [options] filenames
```

which 命令只能定位可执行文件的位置，其搜索只在 PATH 指定的目录内进行，因此不能定位其他类型文件。可用--skip-alias 参数抑制别名的输出。

which 命令使用示例如下：

```
#which which                   #查找命令 which 在 PATH 中的位置
#which ls                      #查找命令 ls
#which --skip-alias ls         #查找命令 ls，不要别名
#which date time
```

2）whereis

whereis 可用于对二进制命令文件、源代码和手册页位置定位。其用法为：

```
whereis [ -bmsu ] file ...
```

参数-b、-s 和-m 分别用于搜索 file 的二进制、源代码和手册页文件，比如：

```
#whereis ls who          #查找命令 ls 和 who 的二进制、源代码和手册页文件位置
#whereis -b ls who       #查找命令 ls 和 who 的二进制文件位置
#whereis -m ls who       #查找命令 ls 和 who 的手册页文件位置
```

3）whatis 和 apropos

whatis 和 apropos 命令用于查找 whatis 数据库。通过 whatis 命令可以查找到各种类已经安装的，如命令、系统调用、库函数、配置文件格式等的摘要信息。如果查找的主题（命令或字符串）能找到，则这些摘要信息可以告诉用户它们是干什么用的和在手册页的什么位置。

说明：whatis 数据库是 makewhatis 工具根据各种已经安装的手册页创建的包含各种手册页摘要或眉批的数据库。

whatis 等价于 man -f，用于从 whatis 数据库中查找完整的单词，属于精确查找，而 apropos 等价于 man -k，用于从数据库中查找字符串，属于模糊查找。

whatis 和 apropos 的使用示例如下。

```
#whatis echo     或    man -f echo
#apropos echo    或    man -k echo
```

7．别名管理（alias、unalias）

1）别名定义（alias）

alias 命令用于为一个可执行程序定义别名。当一个别名定义之后，就可以像命令一样被使用，别名的搜索优先于其他类型的命令。alias 是 bash 的一个内部命令，其用法为：

```
alias [-p] [name[=value]] …
```

参数-p 用于显示别名列表，当不带参数运行 alias 时，将显示当前的所有别名。

alias 的应用示例如下：

```
#alias                   #显示所有已经定义的别名
#alias ll rm             #显示 ll 和 rm 的别名定义
#alias li='ls -l -i '    #定义别名 li，其功能为 ls -l -i
```

2）别名取消（unalias）

unalias 用来取消由 alias 定义的别名，其用法为：

```
unalias [-a] [name ...]
```

选项-a 用于取消所有别名。比如：

```
#unalias ll              #取消别名 ll
#unalias -a              #取消所有别名。
```

8．任意精度计算器（bc）

bc 是一个支持任意精度计算的解释语言，提供交互式操作。其用法为：

```
bc [-l] [file ...]
```

若提供有 file 参数，则 bc 首先读取指定的输入文件 file，其中必须为包含 bc 命令序列、语句或函数定义的文本文件，在输入处理完毕后，继续读取标准输入作为处理和解释对象。若不带参数运行 bc，则它进入交互式界面。

bc 支持类似 C 语言的语法，使用起来较方便，还可通过-l 选项使用函数库。

可用以下三种办法之一退出 bc：quit，halt，Ctrl+D。

1）数

bc 命令允许用户指定十进制、八进制、十六进制或其他进制作为输入和输出进制：ibase=n 指定输入进制，obase=m 指定输出进制，默认值为十进制。bc 命令始终使用.（点）来表示小数点，用户还可用 scale=n 来指定输出数据小数部分的位数为 n。

在十六进制中，A、B、C、D、E 和 F 分别表示十进制的 10、11、12、13、14 和 15。

2）变量

在 bc 中，数可被存入两种类型变量：简单变量和数组变量，数组变量形式为 name[ ]。所有变量的字母都必须是小写的。特殊变量有 ibase、obase、scale 和 last。last 代表最后显示的数据的值。显示一个变量值的办法是输入变量名后直接回车，也可以通过直接输入数值或表达式后直接回车的办法进行直接计算或数制转换（输入和输出进制由 ibase 和 obase 决定）。

3）注释

可以使用 shell 的方式进行注释，也可以使用 C 语言中的“/*… …*/”进行注释，但不能使用//方式。

4）继续行

若一行内容太长，则可以像 C 语言和 shell 一样使用继续行。

5）表达式

可像其他高级语言一样使用表达式，在表达式中可用的运算符如表 3-49 所示。

表 3-49　bc 的运算符

运算符类	运　算　符	意　　义
算术运行符	=	赋值运行符
	+, -, *, /, %, ^	加、减、乘、除、求余和乘方
	++，--	自加和自减运算符
	op=	计算赋值运算，如+=，*=，%=等
	( )	括号
逻辑运算符	<,<=,>,>=,==,!	小于，小于等于，大于，大于等于，等于，不等于
布尔运算符	&&,||，!	与，或，非

6）函数

（1）bc 使用的标准函数有：length(var)（变量的有效长度）、read()（读取一个字符串，如 x=read()）、scale(var)（求一个变量的小数位）和 sqrt(var)（求一个数的开平方）。

（2）库函数（使用-l 参数时）：s(var)（sin）、c(var)（cos）、a(var)（arctan）、l(var)（ln）、e(var)（exp）和 j(n,x)（x 的 n 阶贝塞尔函数）。

（3）自定义函数：用户可以自己定义函数，方法为：

```
define func_name(parameters) {
        auto_list
        statement_list
}
```

7）语句

bc 可使用的语句有：print list（输出列表）；if(expr) statement1 else statement2（if 结构）；while(expr) statement（while 结构）；for([expr1];[expr2];[expr3]) statements（for 结构）；break；continue（循环控制）；halt、quit（退出 bc）；return，return(expr)（函数中返回）；limits（显示 bc 的最大值）等。

8）示例

（1）一般（数学）计算。

```
#bc -l                               #进入 bc，以下为在 bc 环境下的输入
        scale=4                      /*定义小数位数*/
        123+456*850-s(2)^2           /*计算 123+456*850-sin(2)**2 */
#       387722.1734                  /*输出结果*/
```

（2）进制转换。

```
#bc                                  #进入 bc，以下为在 bc 环境下的输入
        obase=16                     /*定义输出进制为 16*/
        32767                        /*结果为 7FFF*/
        123456                       /*结果为 1E240*/
```

（3）使用自定义函数。

```
#bc –l                               #-l 启动使用库函数
scale=2
define myf(x) {                      /*定义函数 myf*/
        auto i,j                     /*定义自动变量*/
        j=1
        for(i=1;i<=x;i++)
                j=i*j
        return (j)
}
myf(5)                               /*计算 myf(5)，结果为：120*/
l(myf(10))+5^6                       /*计算 ln(myf(10))+5^6 结果为：15640.10*/
```

（4）在 shell 中使用 bc 计算。

```
$ echo "scale=10; 4*a(1)" | bc -l             #计算并显示 π 的值
$ pi=$( echo 'scale=10; 4*a(1)' | bc -l)      #计算 π 的值，并赋给变量 pi
$ echo $pi                                    #显示 π 的值：3.1415926532
```

习题 3

1．思考题

（1）shell 的基本功能有哪些？

（2）Linux 系统中的主要目录有哪些？各有什么作用？

（3）工作目录及其父目录可分别用什么表示？

（4）在如图 3-1 所示的目录结构中，若处在 usr1 目录中的用户要访问 include 目录中的 stdio.h，可以采用什么样的路径，对应的带有路径的文件名是什么？

（5）Linux 系统中常用的通配符有哪些？试举例说明它们的作用。

（6）常用的 shell 环境变量有哪些？怎么查询和设置环境变量？

（7）什么是输入/输出重定向？管道的功能是什么？试举例说明如何使用它们。

（8）shell 中的引号有哪几种？各有什么作用？试举例说明之。

（9）什么是参数替换？什么是命令替换？试举例说明如何使用它们。

（10）shell 的种类有哪些？

2．选择题

（1）下列设备属于块设备的是（　　）。

A．键盘　　B．终端　　C．游戏杆　　D．硬盘

（2）cd 命令可以改变用户的当前目录，当用户输入命令“cd”并按 Enter 键后，（　　）。

A．当前目录改为根目录　　B．当前目录不变，屏幕显示当前目录

C．当前目录改为用户主目录　　D．当前目录改为上一级目录

（3）命令 echo -en '\a'的输出为（　　）。

A．响铃　　B．\a　　C．'\a'　　D．a

（4）超级用户的用户主目录是（　　）。

A．/　　B．/root　　C．/home　　D．/home/root

（5）要删除目录/home/user1/subdir 连同其下级目录和文件，不需要交互确认，正确的命令是（　　）。

A．rm -rf /home/user1/subdir　　B．rm -df /home/user1/subdir

C．rmdir -pf /home/user1/subdir　　D．rmdir -P /home/user1/subdir

3．综合题

（1）试述在 Linux 系统中如何进行日期和时间管理。在图形界面下可以实现时间和日期管理吗？若能，如何做？

（2）在 UNIX/Linux 系统中有些操作是有副作用的，如 cp、mv 和 rm 等，应该如何避免？

（3）文件的属性和类型有哪些？如何确定某文件的类型？如何获得文件的属性？

（4）试简述 Linux 系统的信号通信机制。如何查询可用信号？如何实现信号通信？如何捕获信号？

实验 3

1．变量、三种引号及变量替换、命令替换和参数替换。
2．I/O 重定向与管道。
3．目录管理与文件操作。
4．vi 与文本文件操作，试试一个图形界面文本文件编辑器 gedit。
5．环境变量与别名管理。
6．时间与时区管理命令、进程管理基本命令和其他管理命令。

系统管理篇

- 用户、组和密码管理
- 文件系统管理及使用
- 进程、任务与作业管理
- 系统安装、扩充、启动与管理
- 设备管理
- 网络配置、管理与基本应用
- Linux 系统的安全

第 4 章　用户、组和密码管理

4.1　UNIX 系统的用户和组

对 UNIX/Linux 系统的管理与使用是以用户（user）为主体的。自然人要想使用系统，必须以某个已在系统中存在的用户身份登录，且通过密码验证后才能进入系统，并按权限使用。对于系统中已经存在的用户，可以在图形或字符界面下登录，也可通过网络远程登录。

UNIX/Linux 系统中有两类用户：超级用户和一般用户。超级用户有至高无上的权力，可做任何事情；一般用户只能在给定的权限范围内进行工作，不得越权行事。用户又按实际需要分成不同的组（group）。同组用户按规定和目的享有某些共同的权限。

4.1.1　用户与 uid

在 UNIX/Linux 系统中每个用户都有一个用户名（user name），系统还给每个用户分配了一个用户标识（uid：user identification）。用户 uid 是系统辨识用户的唯一标识，而用户名则是用户的外部表示。

用户信息存放在文件/etc/passwd 中。用户标识 uid 可用命令 id 来查询。

4.1.2　组与 gid

系统在创建用户时，为每个用户安排了一个归属组（group）。系统中的每个组都有一个组名（group name）和一个组标识（gid：group identification）。

一个组是具有某种联系或关系的用户集合。比如，在某种业务方面，需要把操作某些数据文件或数据库的用户放在一个组中，以实现数据共享或共同操作。一个组中可以包含多个用户，一个用户可参与不同的组。

组信息存放在文件/etc/group 中，组标识 gid 也可用命令 id 来查询。

4.2　与用户和组管理相关的文件

与用户管理相关文件：/etc/passwd、/etc/shadow、/etc/group、/etc/login.defs、/etc/default/useradd 等，用于对用户设置和登录项目进行控制。

4.2.1　/etc/passwd

/etc/passwd 是系统用户数据库文件，它包括系统内所有已经注册用户的信息。该文件是一个文本文件，它的每一行描述一个用户的信息，为由“:”分隔的 7 个字段。结构为：

```
username:[password]:uid:gid:[comment]:dir:[shell]
```

username：用户名，必须以字符开始，不要使用大写字母。

password：密码。在 Linux 系统中此项一般设置为“x”，用于表示使用影子密码，此时密码在/etc/shadow 文件中。

uid：用户标识号。不同的 UNIX/Linux 系统，该项的取值范围可能不同，在 Linux 系统中该项的取值范围在/etc/login.defs 文件中定义。一般用户 uid 满足条件：UID_MIN≤uid≤UID_MAX。

gid：组标识号。说明同 uid。

comment：说明域。一般包括用户详细说明信息，如用户全称、住址、电话号码等，此项内容中不能包含冒号，但可以为空。

dir：用户家目录（$HOME），即用户登录成功后所在的工作目录。

shell：用户登录成功后所使用的 shell 程序，也可是用户实用程序，当为空时，系统默认/bin/sh。如果想让用户登录后直接执行某个程序，如/usr/bin/my_shell，则可修改此项内容为/usr/bin/my_shell。若此项设置为一个不存在或不可执行的文件，则用户不能成功登录。

以下为/etc/passwd 文件中的部分内容：

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

第一行是 root 用户的信息，用户名为 root，uid 和 gid 均为 0，描述信息为“root”，家目录为/root，使用 shell 为/bin/bash。第二行是 bin 用户的信息。

在/etc/passwd 内有很多用户，事实上它包含了系统内的所有用户。用户有很多类型，设置用户也有不同目的。有的是真实用户，可以用来登录系统，而有的则不能，但可以它的身份完成指定任务。这种不能用于登录的用户被称为伪用户，伪用户的登录 shell 一般为 nolgin。例如，bin 用户拥有可执行的命令文件；daemon 用户拥有系统服务进程；lp 用户被打印机子系统使用。

4.2.2 /etc/shadow

/etc/shadow 是影子密码文件。当系统启用影子密码时，用于存放系统内用户加密后的密码和用户登录控制信息。其结构为：

```
username:password:lastchanged:min:max:warn:inactive:expired:resv
```

username：与 password 内容对应的用户名。

password：加密后的密码。若为空表示无密码，开头的两个字符为“!!”时表示用户被上锁，现在不可使用。一般情况下，新创建的用户是被上锁的。

lastchanged：最后修改日期。从 19700101 到上次修改密码的天数间隔。

min：必须修改密码的剩余天数。

max：密码的最长有效天数。

warn：更换前警告的天数。

inactive：账号被取消激活前的天数。

expired：账号不活跃的天数。

resv：保留。

用户一般情况下不需要直接操作/etc/shadow 文件。

4.2.3 /etc/group

/etc/group 是组定义文件。它是一个文本文件，每行描述一个组。结构为：

```
groupname:password:gid:userlist
```

groupname：组名。

password：组密码。如果此域为空，则不使用密码。在 Linux 系统中设置为“x”。可用 gpasswd 设置此密码，并由 newgrp 等命令使用。

gid：组标识号。

userlist：组成员，为以逗号分隔的用户名。

以下是/etc/group 文件的一行：

```
bin:x:1:root,bin,deamon
```

说明：组名为 bin，gid 为 1。组员有 root、bin 和 deamon。

4.2.4 /etc/login.defs

/etc/login.defs 文件定义了与用户创建和密码管理相关的常量值。在红帽系统中，其部分内容如下：

```
#grep -v -E  '(^#)|(^[[:space:]]*$)' /etc/login.defs
 MAIL_DIR       /var/spool/mail                 #mail 目录
 PASS_MAX_DAYS        99999                     #密码可用的最大天数
 PASS_MIN_DAYS            0                     #两次修改密码之间的最小天数
 PASS_MIN_LEN             5                     #密码最小长度
 PASS_WARN_AGE            7                     #密码到期前给出警告的天数
 UID_MIN               1000                     #UID 的最小值
 UID_MAX              60000                     #UID 的最大值
 GID_MIN               1000                     #GID 的最小值
 GID_MAX              60000                     #GID 的最大值
 CREATE_HOME            yes                     #在创建用户时，默认创建家目录
```

由文件的内容可知，在创建用户时默认将创建家目录，因为 CREATE_HOME 为 yes。若其值为 no 则默认不创建，尽管已经指定了家目录。

但在 Ubuntu 中，此文件中没有包含“CREATE_HOME …”，意味着“CREATE_HOME no”，因此在 Linux 系统下，创建用户时默认不为其创建家目录。

4.2.5 其他文件

在不同 UNIX/Linux 系统的实现中，所使用配置与管理文件可能是不同的。Linux 中创建用户相关的文件有，/etc/default/useradd、/etc/skel 和其他相关的配置文件。/etc/default/useradd 是创建用户的默认属性参考值，而/etc/skel 是一个框架目录，其中的内容都是隐藏的，当新

用户创建后要将其内的文件和目录复制到新用户的家目录，并重新设置用户主、用户组和权限，以作为新用户从不同界面登录的默认配置。

另外 Linux 系统提供有 pwck 和 grpck 用于对 password、group 和 shadow 文件间关系正确性有检查。

还有 vipw 和 vigr 命令用于对 password、shadow 和 group 文件的编辑。在编辑过程中，它们通过锁控方式保证被编辑文件不会被破坏，因此要比直接使用 vi 等文本编辑器编辑这些文件来得安全。其用法为：

```
vipw [options]
vigr [options]
```

选项-p 和-g 分别用于编辑 password 和 group 文件；-s 用于编辑与它们对应的 shadow 和 gshadow 文件。

4.3 用户管理命令

用于用户管理的命令有 useradd、usermod 和 userdel，它们分别用于用户创建、用户属性修改和用户删除。

4.3.1 增加用户（useradd）

1．功能与用法

命令 useradd 的功能是创建新用户，其用法为：

```
useradd [options] user                #添加用户
useradd -D                            #显示默认值
useradd -D [options]                  #设置默认值
```

2．参数说明

useradd 的部分参数如表 4-1 所示。

表 4-1　useradd 的部分参数

选　　项	功 能 描 述
-c comment	新用户注释信息。不含“:”和回车的长度小于 512 的字符串
-d home_dir	新用户家目录。默认时，以新用户为名在/home 下创建家目录
-D	显示新创建用户时的参数默认值，或为新用户设置新的参数默认值
-e expire_date	以 MM/DD/YYYY 格式设置账号到期日
-f inactive_days	密码到期后，直到账号永久停止使用时所经过的天数，默认为-1，表示不启用该选项
-g group	设置新用户所属的基本组
-G group1，group2，…	将新用户添加到由逗号分隔的组列表中，一个用户可同时归属多个组
-k skel_dir	设置包含初始配置框架文件的目录，默认框架目录为/etc/skel。该选项只能与-m 一起使用

（续表）

选　　项	功 能 描 述
-m	创建用户家目录，并将框架目录的默认配置文件复制到该目录中
-M	创建新用户，但不创建用户家目录
-n	关闭创建新用户时，创建同名组的特性
-o	允许使用-u uid 指定一个与已经存在用户相同的 uid，即允许不同名的用户具有相同的 uid
-r	此为 Redhat Linux 新增选项，用于创建系统用户
-s shell	指定用户登录 shell，若不指定则采用系统默认 shell:/bin/bash
-u uid	为用户指定 uid。若不指定系统将自动分配一个。若 uid 已经被使用，则必须与-o 配合使用

3．关于创建用户的说明

（1）若不带选项运行 useradd，如 useradd newuser，则使用默认方式创建 newuser。若在创建用户时没有创建家目录，则为保证 newuser 创建后可以登录还必须手工创建 newuser 的家目录，并修改家目录归属关系和存取权限。

（2）创建用户时，是否创建用户家目录，默认情况下取决于/etc/login.defs 中变量 CREATE_HOME 的值，但可以使用-m 强制创建，也可以使用-M 强制不创建。不创建用户家目录也是常见的，当多个用户共用一个家目录时，只要第一个用户创建时创建了家目录就可以，而以后用户直接使用已经存在的目录。这时应该注意目录的存取权限问题，必要时可将这类用户放在一个组中，但要求目录及内容能被同组人按特定权限访问。

（3）若未指定密码，则在一切准备好后，还必须经超级用户为新建用户设置或修改密码后才能正常登录和使用。

（4）Linux 默认，在创建新用户时，按新用户名创建一个新组，该组为新用户的归属组，或称为基本组。如果想让新建用户使用一个已经存在的组，则可使用-g 参数；如果同时还想让新建用户也归属到其他组，则可以使用-G 参数。

4．创建用户示例

（1）以默认方式创建用户

```
#useradd test1                    #以默认方式创建用户 test1。之后使用以下命令观察文件变化
#grep test1 /etc/passwd /etc/group /etc/shadow /etc/gshadow
##发现在文件/etc/passwd、/etc/group、/etc/shadow 和/etc/gshadow 中分别添加了如下行：
 /etc/passwd:     test1:x:1000:1000::/home/test1:/bin/bash
 /etc/group:      test1:x:1000:
 /etc/shadow:     test1:!!:13563:0:99999:7:::        #!!表示用户未启用或被上锁
 /etc/gshadow:    test1:!::                          #!表示未启用组密码
```

##/etc/shadow 文件中 test1 项中的密码项内容为“!!”，是一个上锁状态，说明此用户还没有被启用（没有密码且不能登录），只有在超级用户为它修改密码或解锁后才能被启用。

（2）以非默认方式创建用户。

```
##创建一个名为 test2 的用户，且为其创建家目录/usr/test2，而非默认的/home/test2
#useradd –d /usr/test2 –m test2
```

（3）创建用户 test3，让其与 test1 使用相同的家目录和相同的组。

```
#useradd -d /home/test1 -g test1 -M test3        #不创建家目录
```

（4）创建用户 test4，让其与 test1 具有相同的 UID、家目录和组。

```
#useradd -u `id -u test1` -o -g test1 -d /home/test1 test4
```

使用非默认模式创建用户时，在创建或建后使用时可能会遇到一些问题。例如，若指定的目录新用户不可访问或没有创建，则创建后不能登录；若指定的 uid 不合适，则可能会创建失败。

4.3.2 用户删除（userdel）

对于系统中不再使用的用户账号，应尽早删除它，因为多余用户的存在可能是安全隐患。删除用户的命令是 userdel。其用法为：

```
userdel [-r] user
```

选项-r 的作用是在删除用户的同时删除用户的家目录。

关于用户删除有以下说明：

（1）在某些高级别的 UNIX 系统里，用户一旦被创建，是不能随便从系统中删除的，用户的 uid 也不能被再重新使用，但可以通过过期或上锁的办法将它闲置起来。

（2）对于已经登录的用户或有属于该用户程序在运行的用户来说，删除可能失败。

（3）一般情况下删除用户时只删除/etc/passwd，/etc/shadow，/etc/group 及相关文件中与该用户相关的内容，而不删除用户的家目录，因为用户家目录中可能存放重要数据。对用户家目录的删除应由用户或经用户确认后由管理员来进行。

（4）若要在删除用户的同时删除用户家目录，可使用-r 选项。

4.3.3 用户修改（usermod）

当用户被创建之后，它的属性可通过命令 usermod 来修改。usermod 的用法为：

```
usermod [options] [-l newname] [-L|-U] user
```

usermod 的多数选项与 useradd 的选项相同且意义相同（参见表 4-1）。新增的选项有：-l newname：设置用户新名字，将用户名由原来的 name 变为 newname。-L：将用户密码上锁。-U：将用户密码解锁。

4.4 组管理命令

用于组管理的命令有 groupadd、groupmod 和 groupdel，它们分别用于组的创建、修改和删除。

4.4.1　组创建（groupadd）

1．功能及用法

Linux 默认，在创建用户时创建一个与用户名同名的组，如果想改变这种默认，使新用户所在的组为某个已经存在的组，则可以在创建用户时使用-g group 指定一个已经存在的组。一个已经存在的组可能是由 useradd 在创建用户时自动创建的，也可能是使用 groupadd 创建的。

groupadd 命令的功能是创建新组，其用法为：

```
groupadd [-g gid [-o]] [-r] [-f] group
```

2．参数与说明

groupadd 命令的参数如表 4-2 所示。

表 4-2　groupadd 命令的参数

选　　项	功 能 描 述
-f,--force	当指定组已经存在时，不报告错误而直接成功返回
-g gid	指定新创建组 gid，gid 应该满足条件：GID_MIN≤gid≤GID_MAX
-o	允许使用-g gid 指定一个与已经存在组相同的 gid
-r	创建系统组（0≤gid<GID_MIN），如果不用-g 指定 gid，则从可用 gid 中选择最小的一个

4.4.2　组删除（groupdel）

命令 groupdel 用于删除系统中已经存在的组，其用法为：

```
groupdel group
```

groupdel 不能删除系统中仍然存在的用户的基本组，也就是说只有先删除用户，才能删除它所拥有的基本组。在有的系统中，如果组内成员不空，也不能被删除。

4.4.3　组修改（groupmod）

命令 groupmod 用于修改系统中已经存在组的属性，其用法为：

```
groupmod [-g gid [-o]] [-n newgroup] group
```

当使用-n 时，该命令的作用使得原来的组 group 变为新的组 newgroup，相当于组的更名。当使用-g gid 时，用于更改组的 gid，若指定的 gid 已经被使用，则必须使用-o 告诉 groupmod 这是一个旧组号，或一个已经存在的组。

4.5 密码管理

4.5.1 密码管理综述

UNIX/Linux 系统的标准安全级别为 C2 级,具有用户身份认证、访问控制和操作的可靠性等特点。密码管理是实现身份认证的基础。

可用于密码的字符是广泛的，视不同系统而定。一般来说，可以使用大小写字母、数字、标点符号等，有的系统甚至可以使用光标键和功能键等。密码的长度也视具体系统或安全级而定。使用密码时应注意以下几个问题。

（1）密码应该按规定定期或不定期修改。

（2）密码内不应包含完整的单词、生日、电话号码、姓名、用户名、组名、宠物和地址等信息。

（3）不同系统和用户应该有不同的密码。

（4）密码应易记且不要写在纸上。

（5）要保密，不要共用密码。

（6）输入密码时不要让别人看见，也不要窥视别人的密码。

系统的安全与方便是矛盾的，不能因方便而忽略安全问题，也不能因过于强调安全而影响正常使用，应该在安全和方便之间找到平衡点或进行取舍。

4.5.2 密码管理命令（passwd）

1．功能与用法

命令 passwd 的功能是密码管理，包括改变或删除用户密码，为用户上锁或解锁，改变或显示用户的密码属性等。其用法为：

```
passwd [-k] [-l] [-u [-f]] [-d] [-n mindays] [-x maxdays] [-w warndays][-i inactivedays] [-S] [username]
```

2．参数说明

username 为用户，root 用户可以为指定用户设置密码，若 username 未指定，则为使用 passwd 命令者自己修改密码。passwd 命令的可用参数如表 4-3 所示。

表 4-3 passwd 命令的部分参数

选　　项	功 能 描 述
-d	删除指定用户密码。被删除密码的用户在登录时不需要密码
-i inactivedays	设置用户密码不活动的天数，只有 root 可以使用此选项
-k	告诉系统用户更改密码应在密码到期前进行
-l	对指定用户上锁，只有 root 可以使用此选项

（续表）

选　　项	功 能 描 述
-n mindays	设置用户密码的最短有效期，即允许更改前的天数，只有 root 可使用此选项
-S	报告指定用户的密码状态信息
-u	解锁一个被-l 上锁的用户
-w warndays	设置用户密码到期前提前警告的天数，只有 root 可以使用此选项
-x maxdays	设置用户密码的最长有效期，即需要更改的天数，只有 root 可以使用此选项

除 root 外，修改密码时，系统将提示用户先输入旧密码；密码在到期前是有效的，若在有限时间内没有设置或修改密码，则到期后的下一次必须修改密码；一个用户可被上锁，上锁后的用户一经退出将不能再登录；只有 root 能够上锁或解锁一个用户；尽管密码可以被删除或置空，但不推荐这样做。

4.5.3　密码管理示例

```
#passwd                    #为用户自己修改密码
#passwd test1              #为用户 test1 修改密码
#passwd -d test1           #为用户 test1 删除密码
#passwd -l user            #对用户 user 上锁。上锁后用户不能再登录系统
#usermod -L user           #同上
#passwd -u user            #对用户 user 解锁。解锁后用户可以再次登录系统
#usermod -U user           #同上
#passwd -x 2 test2         #设置用户 test2 密码最长有效期为 2 天
```

4.6　用户、组和密码管理图形界面

这里只介绍“设置中心（settings）”中的用户设置，启动方法是：

```
“设置中心（setting）”→“用户（Users）”    # 或在 CLI 界面执行
# gnome-control-center                       # 红帽：CLI 界面
# unity-control-center                       # Ubuntu：CLI 界面
```

可以进入如图 4-1 所示的用户管理图形界面，操作者可以进行的操作有用户添加（“+”）、用户删除（“−”）和用户属性修改。需要说明的是，当操作者权限不够时，需要单击“ Unlock ”输入操作者密码获得操作权之后才可进行相关操作。

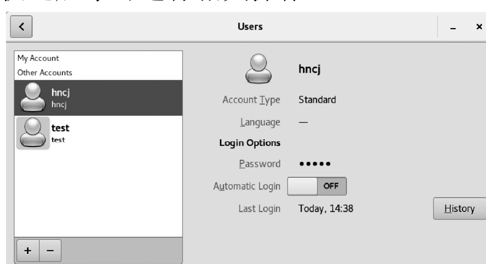


图 4-1　用户管理图形界面

4.6.1　添加用户

单击图 4-1 所示左下角的“+”，可以添加用户，添加用户界面如图 4-2 所示，操作过程参见用户属性修改。

图 4-2　添加用户界面

4.6.2　用户属性修改

当从图 4-1 左侧的用户中选择一个（hncj）用户时，右侧内容所选用户的属性，操作者可以根据需要修改其中一项或多项。可修改的内容如下。

（1）图标。一个标识用户特征的图片。可以是系统中的已有图片，也可以是自己的照片。

（2）用户（User）。符合规定的用户名。

（3）用户类型（Account Type）。可以是标准（Standard）的或管理员（Administrator）。若设为管理员，则系统将该用户名加入 wheel 组，在 ubuntu 中将其加入 adm 和 sudo 组。

（4）语言（Language）。用户要使用的语言，可以根据需要设置。

（5）密码（Password）。当选择修改密码时，会弹出如图 4-3 所示的界面，给用户两个选择：一是允许用户在下次登录时自己设置密码（Allow user to set a password when they next login）；二是现在就要设置密码。当选择一时，用户在下次登录时，要自行设置密码。当选择二时，要输入两次相同的符合安全要求的密码，当“修改（Change）”按钮变得可用时，单击之修改密码。在修改密码时，还可以单击密码输入框右端的“”按钮为用户生成一个随机密码。

（6）自动登录（Automatic Login）。这是一个开关，可以打开，也可以关闭，但系统中只能有一个用户是自动登录的。当一个用户设有此属性时，在下次开机时，将自动登录到该用户。



图 4-3　密码设置界面

4.6.3　用户删除

当一个用户被选中后，可以单击左下角排列第二的“-”按钮删除之。在真正执行用户删除之前，系统会给出如图 4-4 所示的提示界面。

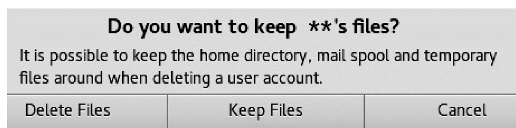


图 4-4　用户删除选项

可以单击“Cancel”撤销用户删除操作。若选择了“Delete Files”，则将在删除用户的同时删除用户的家目录；若选择了“Keep Files”，则只删除用户而不删除用户家目录。

在以上操作过程中，没有涉及组的处理。若要同时处理组，在红帽系统中可使用“system-config-users”。

4.7　与用户身份和位置相关的其他命令

在 UNIX/Linux 系统中还有一些命令可用于确定或改变用户的位置与身份，这些命令是 id、who、whoami、write、wall、tty、su 和 sudo 等。

4.7.1　显示与用户和组相关的身份信息（id）

1．功能与用法

id（identification）命令用于显示用户的 uid、gid 及相关信息。其用法为：

```
id [options] [username]
```

2．参数说明

id 命令的常用参数如表 4-4 所示。

表 4-4　id 命令的常用参数

选　　项	功 能 描 述	选　　项	功 能 描 述
-a	显示相关的所有 id 信息	-n	以名字方式显示信息，与-u，-g，-G 配合使用
-g	显示用户有效 gid	-r	显示真实 uid 或 gid，与-u，-g，-G 配合使用
-G	显示指定用户所归属的其他组 gid	-u	显示有效 uid，而非真实 uid

3．id 使用示例

```
#id -G –n                   #以组名方式显示自己的 gid 信息
#id ftp                     #显示指定用户(ftp)的 id 信息
```

4.7.2　显示已登录用户的信息（who）

1．功能与用法

who 命令用于显示系统中已登录用户的信息。其用法为：

```
who arg1 arg2
who [options] [FILE]
```

当以第一种格式使用 who 命令时，相当于 who am i，总是显示命令使用者的信息。在文件/var/log/wtmp 和/var/run/utmp 内容中记录有系统运行过程中用户登录和工作信息，who 命令可以使用它。若不指定 FILE 文件，则 who 命令使用这两个文件。

2．参数说明

who 命令的常用参数如表 4-5 所示。

表 4-5　who 命令的常用参数

选　　项	功 能 描 述	选　　项	功 能 描 述
-a	显示所有信息	-p	显示由 init 派生的活动进程
-b	显示系统最后一次启动时间	-q	显示已登录的所有用户名和用户数
-d	显示僵尸进程信息	-r	显示系统的当前运行级别
-H	显示标题头信息	-s	以默认格式显示
-t	显示系统时钟最后一次修改时间	-T, -w	显示用户的状态信息：+，-或?
-m	只显示与所使用终端相关的信息	-u	显示用户

3．示例

```
#who                  #显示所有正在系统中工作的用户信息
#who -H               #显示所有正在系统中工作的用户信息，并显示标题
#who am i             #显示操作者自己的信息
#who -r               #显示系统当前的运行级别
```

4.7.3　显示使用者的用户名（whoami）

whoami 命令用于显示与当前有效 id 相关的用户名，也就是显示使用者自己的用户名，功能相当于 id -un。

4.7.4　向系统中的指定用户发信息（write）

write 命令用于向系统中的指定用户发送指定信息，其用法为：

```
write user [ttyname]
```

user 为系统中的用户名，ttyname 为用户所使用的终端设备名。当向系统中同名（多处同时登录）用户之一发信息时，需要指定终端名。

若使用输入重定向，则从输入文件中读取信息，发送完毕后结束；否则从终端键盘逐行读取，直到用户输入^D 结束工作。如果对方也使用 write 向自己发信息，则两者可以“聊天”。

4.7.5　向系统中已登录的所有用户发信息（wall）

wall（write to all）命令的功能是向注册到系统中工作的所有用户发送广播信息，所发信息将被显示在每个用户的终端屏幕上。wall 的用法为：

```
wall [file]
```

file 为事先准备好的文件。当不带参数运行时，wall 将使用标准输入，用户在输入完毕时应按^D 结束。系统允许发送的信息最多为 20 行，长行或将被截断。当用户使用已经准备好的文件内容作为信息发送时，可以使用输入重定向。Fedora 还允许 wall 将其命令行参数的内容作为信息发送。

wall 的工作可能会受终端接收信息状态的影响（参见 mesg 命令），接收信息被禁止的用户终端将不能收到广播信息。但是不论如何，当超级用户或管理员用户使用 wall 时，每个终端都将收到广播信息，同名用户也可以收到同名用户发送的信息。

wall 的广播信息可能会扰乱接收者的工作屏幕，更有甚者，当接收的终端正处在透明打印时，收到广播信息将会送到终端打印机而造成真正的干扰。其他类型的广播或向指定终端的重定向输出信息也会造成类似干扰，请使用时小心。

4.7.6　确定用户所使用的终端设备（tty）

tty 命令用于显示命令自己使用的终端设备，以确定使用位置，其用法为：

```
tty   [-s]
```

若不使用选项-s，则在命令所用终端上显示所用的终端名。当使用-s 时，什么也不输出，只返回状态。状态为 0 时，表示命令是在终端上运行的，否则表示是在后台等非终端环境下运行的。

4.7.7　不退出系统而将自己切换成其他用户（su）

1．功能与用法

su 命令的功能是在使用者不退出系统的情况下以其他新用户的身份工作。当不指定任务时，将以新用户的身份启动一个 shell；当指定任务时，则以新用户的身份执行任务。在以新用户工作的过程中，有效 uid 和有效 gid 变为新用户的 uid 和 gid。su 命令的用法为：

```
su    [options] [-] [newuser [args]]
```

2．参数说明

su 命令的参数说明如表 4-6 所示。

表 4-6　su 命令的常用参数说明

选　　项	功 能 描 述
-，-l，--login	以新用户身份启动一个 shell，并以新用户环境进行工作
-c CMD，--command=CMD	以新用户身份和环境执行命令 CMD
-m，-p	用户切换时，不重新设置环境变量
-s shell，--shell=shell	切换时指定 shell。指定的 shell 要在文件/etc/shells 中有定义

若不指定用户名，则默认为 root。当以新用户身份工作完毕后，变回到原来的自己。除非由 root 向其他用户切换，否则都要进行密码验证。

以 su newuser 方式切换用户，将使用原先的环境，这就形成了新用户使用老环境的问题，可能不配套。而以 su - newuser 切换的方式，将使用新用户 newuser 的新环境，犹如退出后重新登录一样。

如果要在批处理程序内以另一个新用户 newuser 直接执行某程序 command，则可使用下列形式：

```
su - newuser    -c "command args"                    或
su - newuser    --command="command args"
```

在操作系统启动期间，常以这种方法启动某服务程序。

3．应用示例

```
#su newuser                         #切换到用户 newuser 后使用原环境工作
#su - newuser                       #切换到用户 newuser 后使用新用户环境工作
#su - newuser -c "command args"     #以 newuser 身份执行"command args"
##以 postgres 身份登录，使用/bin/sh 执行命令/usr/bin/initdb
#su - postgres -s /bin/sh -c "/usr/bin/initdb --pgdata=/var/lib/pgsql > /dev/null 2>&1"
```

4．安全控制及说明

在 Linux 系统中，su 的使用都受 PAM 的控制。在红帽系统中，PAM 的控制文件为

/etc/pam.d/su，在有的系统中还有/etc/pam.d/su-l 文件专用于对 su -l 的控制，不过/etc/pam.d/su-l 又包含了/etc/pam.d/su，关键控制部分仍在/etc/pam.d/su 中。在/etc/pam.d/su 中，用于使用 su 向 root 切换的控制部分如下：

```
#Uncomment the following line to implicitly trust users in the "wheel" group.
#auth       sufficient   pam_wheel.so trust use_uid
#Uncomment the following line to require a user to be in the "wheel" group.
#auth       required     pam_wheel.so use_uid
```

都是被注释掉的没有发挥作用，所有用户均可切换到 root 用户，但需要 root 用户的密码认证。如果（第 2 行）

```
auth  sufficient   pam_wheel.so trust use_uid
```

起作用，则 wheel 组内的用户使用 su 向 root 切换时不需要密码，而其他用户也可以向 root 用户切换，但需要 root 的密码。如果（第 4 行）

```
auth  required     pam_wheel.so use_uid
```

起了作用，则只有 wheel 组内的用户才能使用 su 向 root 切换，且需要 root 密码，其他用户则不能向 root 用户切换。

4.7.8　以其他用户身份执行程序（sudo）

1．功能及用法

su 命令告诉我们，用户可以在不退出系统的情况下将自己变换为其他用户，或以其他用户的身份工作。sudo 也允许用户以超级用户或其他用户的身份执行命令，其用法为：

```
sudo -h|-k|-l|-v|-V
sudo [-bH]|[-p str] [-u user] [-g group] {-s|cmd}
```

2．参数及说明

sudo 的常用参数如表 4-7 所示。

表 4-7　sudo 的常用参数

参　数	意　　义	参　数	意　　义
-b	后台执行命令	-h	帮助
-l	列出自己使用 sudo 可以执行的命令	-U user	与-l 配合，列-U 所指定 user 可以执行的命令
-H	重置环境变量 HOME（默认不修改 HOME）	-p str	更改密码提示符。%u 为用户名，%h 为主机名
-u user	指定用户名或 uid，默认为 root	-g group	指定组名或 gid
-k	强迫下次执行 sudo 时输入密码	-v	延长密码有效期 5 分钟
cmd	指定将要执行的命令	-s	执行 SHELL 指定的 shell

3．配置文件 sudoers

/etc/sudoers 是 sudo 的配置文件，能使用 sudo 命令的用户必须在此文件中定义。系统还为此配置文件提供了一个专用的编辑命令 visudo，除了编辑修改 sudoers 文件外，还可帮助使用者检查配置文件内容的正确性。修改 sudoers 文件时请使用 visudo，但不同系统中使用的编辑器不同，Fedora 使用 vi。

在 sudoers 文件，空行和以“#”开头的行为注释，有效行的结构为：

```
USERs HOSTs=(RUNASs) COMMANDs    #意为：哪些人在哪里=(以什么身份)   干什么
```

简单示例如下：

```
root ALL=(ALL) ALL                    #root 可在任何位置以任何人的身份做任何事
%sudo ALL=(ALL) ALL                   #sudo 组中的用户可在任何位置以任何人的身份做任何事
%wheel ALL=(ALL) ALL                  #wheel 组中的用户可在任何位置以任何人的身份做任何事
zh3 ALL=(ALL) /sbin/shutdown #zh3 可在任何位置以任何人的身份执行/sbin/shutdown
```

有关配置文件/etc/sudoers 的更多说明请参阅在线文档或其他文献。

4．示例

设当前用户为 test，在默认情况下，首次执行 sudo 命令，以以下过程介绍 sudo 的使用。

（1）检查用户可以使用 sudo 执行的命令。

```
$ sudo -l                    #首次执行时
    We trust you have received the usual lecture from the local System
    Administrator. It usually boils down to these three things:
        #1) Respect the privacy of others.
        #2) Think before you type.
        #3) With great power comes great responsibility.
    [sudo] password：******
    Sory,user test may not run sudo on localhost. #或
    test is not in the sudoers file. This incident will be reported.
```

再次运行：

```
$ sudo -l                         #再次执行时，直接报错
```

（2）以 root 或管理员身份修改 sudoers 文件，使 test 能做任何事

```
#visudo                           #执行 visudo 命令，在 sudoers 最后增加如下行
test    ALL=(ALL) ALL             #注意：这样，test 的权限就太大了，也可以按以下方式处理
#usermod    -G wheel test         #将 test 添加到 wheel 组
```

（3）再次检查用户可以使用 sudo 执行的命令。

```
$ sudo -l                         #第 3 次执行时
    User test may run the following commands on ******:
        (ALL) ALL                 #说明 test 用户可以以任何身份做任何事
```

习题 4

1．思考题

（1）为了提高系统的安全性，对用户密码应如何管理？
（2）叙述文件/etc/passwd、/etc/shadow 和/etc/group 文件的作用及其结构。
（3）为何要上锁一个用户？如何锁定一个用户？当需要时如何解锁？
（4）如何在删除一个用户时也同时删除它的家目录，在做操作时应注意些什么问题？
（5）如何观察当前系统的运行级别？
（6）如何确定用户所使用的终端？

2．单项选择题

（1）在 UNIX/Linux 系统添加新用户的命令是（　　）。
A．groupadd　　B．usermod　　C．userdel　　D．useradd
（2）添加用户时使用参数（　　）可以指定用户目录。
A．-d　　B．-p　　C．-u　　D．-c
（3）用户 mytest 修改自己的密码可使用（　　）。
A．passwd　　B．passwd -d mytest　　C．passwd mytest　　D．passwd -l
（4）解锁一个用户的命令和选项对为（　　）。
A．passwd -u　　B．passwd –x　　C．passwd -d　　D．passwd -l
（5）删除一个用户密码所使用的命令和选项对为（　　）。
A．passwd -u　　B．passwd -x　　C．passwd -d　　D．passwd -l

3．综合题

（1）试述 Linux 系统的访问控制机制。
（2）试述 su 和 sudo 命令的用法。
（3）试述如何进行用户和组管理。

实验 4

1．用户管理。
2．密码管理。
3．用户管理图形界面。
4．相关管理命令 who、id、tty、wall、su 和 sudo。

第 5 章　文件系统管理及使用

5.1　文件系统权限及管理

操作系统中，与管理文件有关的软件和数据称为文件系统，它负责为用户建立、删除、存取、修改、复制等文件操作工作。

UNIX/Linux 的文件系统除具有一般文件系统特点外，还具有以下特点。

（1）文件是无结构的字符流。

（2）文件可通过设置权限的办法加以保护。

（3）外部设备都被看成文件，在系统内与普通文件统一管理。

UNIX/Linux 用文件存取控制表来解决存取权限的控制问题。存取控制表以文件为单位，把用户按某种关系分为若干类，同时规定每类用户的存取权限。每个文件都有一张存取控制表，该表存放在文件说明中，也就是 i 节点中。

5.1.1　三种权限

UNIX/Linux 系统中，普通权限共有三种。

（1）读权限（r）：指用户对文件或目录查看权限。若用户对某文件或目录不具有读权限，则不能查看其内容。

（2）写权限（w）：指用户对文件或目录的写权力。若用户对某文件没有写权限，则不能修改它；若用户对某目录没有写权限，则不能在该目录内创建文件或子目录。

（3）执行权（x）：指用户对文件的执行权或对目录的进入权限。若用户对某文件没有执行权则不能执行它；若用户对某目录没有执行权则不能进入它。

5.1.2　三类人

UNIX/Linux 系统对文件操作的三种权限只对三类人分配，分别是用户主（user：u）、同组人（group：g）和其他人（other：o）。用户主是文件的拥有者；同组人是与文件主同组的用户；其他人是指除用户主和同组人以外的用户。

5.1.3　权限控制

1．权限的字符串表示

设命令 ls -l -d /dev/sda /dev/tty /boot /bin/bash 的输出中有以下内容：

```
-rwxr-xr-x.   1   root   root 1086480   Sep 30 19:09   /bin/bash
dr-xr-xr-x.   6   root   root      3072   Oct 18 23:04   /boot
lrwxrwxrwx.   1   root   root          3   Mar   3 22:09   /dev/cdrom -> sr0
```

```
brw-rw----.  1  root  disk     8, 0  Mar  3 22:09  /dev/sda
crw-rw-rw-.  1  root  tty      5, 0  Mar  3 22:09  /dev/tty
```

输出的第一列为文件的类型和权限，除第一个字符外，其余为文件的权限描述。例如，目录文件/boot 的权限为 rwxr-xr-x。将此权限字符串从左至右每 3 个分为一组，分别代表文件主、同组人和其他权限，它们分别如下。

rwx：文件主的权限，可读可写可执行。

r-x：同组人的权限，可读可执行，但不可写。

r-x：其他人的权限，可读可执行，但不可写。

就某个文件而言，它对某类用户的权限分配有 3 种：

（1）读权限。若用户对文件或目录有读权限，则对应位置为 r，否则为-。

（2）写权限。若用户对文件或目录有写权限，则对应写权限的位置为 w，否则为-。

（3）执行权限。若用户对文件有执行权或对目录有进入权，则对应执行权限的位置为 x，否则为-。

对于某个文件来说，它的权限按所有者、同组人和其他人从左到右的顺序排列。例如，若用户主的权限为 rwx，同组人的权限为 r-x，其他人的权限为---，则它的权限为 rwxr-x---。

2．权限的数字表示

在数字方式表示下，某类用户对文件或目录的访问权限可用一个三位二进制数来表示。在字符串表示时，若具有某个权限，则对应位为 1，否则为 0，这样某类用户的权限就可表示为三位二进制数，其值为不大于 7 的整数，因此可作为八进制数。例如，目录/dev 的权限为 rwxr-xr-x，它的主、组和其他人的权限分别为 rwx、r-x 和 r-x，表示为二进制数为 111、101 和 101，分别转换为八进制数则为 7、5 和 5。此时，可以说/dev 的权限为 755。此文件的权限如表 5-1 所示。

表 5-1　某文件的权限表示

用户 / 权限 / 表示法	User			Group			Other		
符号表示	r	w	x	r	-	x	r	-	x
数字表示	1	1	1	1	0	1	1	0	1
	7			5			5		

5.1.4　默认权限与 umask

在 UNIX/Linux 系统中，当用户创建文件或目录时，将为它们设置默认权限。文件或目录的默认权限由文件权限掩码 umask 来控制。默认情况下，root 的 umask 值为 0022，普通用户的 umask 值为 0002。文件创建掩码 umask 的值同文件权限一样，可以表示为八进制数或符号串方式。新建文件和目录的默认权限如表 5-2 所示。

表 5-2　umask、文件和目录的默认权限

	root	普通用户
umask	u=rwx,g=rx,o=rx / 022	u=rwx,g=rwx,o=rx / 002
文件	rw-r--r-- / 644	rw-rw-r-- / 664
目录	rwxr-xr-x / 755	rwxrwxr-x / 775

5.2　权限管理命令

5.2.1　设置文件创建掩码（umask）

1．功能及用法

umask 的功能是设置或查询 umask 的值。

创建文件时，系统将用八进制数 777 与文件创建掩码的八进制数按位做减法运算，所得的三位八进制数作为新文件的存取权限。若 umask 的值为 022，则创建“文件”的理论权限为 777−022=755，即 rwxr-xr-x。这对于目录来讲是合适的，可以保证每个用户都可以进入，但是对于一个文件来讲就不一定合适了，因为一个文件不一定具有可执行的内容，因此还要去除各类用户的执行权，故一般文件的权限为 644，即 rw-r--r--。在编程时，新创建文件的权限与 umask 的关系也可这样描述：新创建文件的权限为 0666&~mask，新创建目录的权限为 0777&~mask。

umask 的用法为：

```
umask [ -p ] [ -S ]   [ new_umask ]
```

2．参数及说明

umask 的参数如表 5-3 所示。

表 5-3　umask 命令的参数

参　　数	功　　能
无	显示当前文件创建掩码 umask 的值
new_umask	将新值 new_umask 设置为当前文件创建掩码
-p	当 new_umask 忽略时，显示一个可重用 umask 命令，输出为 umask umask_value
-S	以符号方式显示当前 umask 值

在设置文件创建掩码时，若 new_mask 首位为数字，则用八进制方式解释其内容，否则使用符号方式处理。

3．使用示例

```
#umask                          #显示当前的 umask 值
#umask 0022                     #设置 umask 值为 0022
#umask u=rwx,g=rx,o=            #设置 umask 值为 0027
```

5.2.2　改变文件的权限（chmod）

1．功能及用法

chmod（change mode）的功能为改变文件或目录的访问权限。在设置或改变文件或目录权限时，可以使用符号方式或数字方式。超级用户或用户主可以执行该命令。其用法有以下两种形式。

（1）符号方式：

```
chmod [options] <a|u|g|o> <+|-|=> <str_perm>,… file …
```

（2）数值方式：

```
chmod [options] num_perm file …
```

2．参数说明

chmod 的部分参数如表 5-4 所示。

表 5-4　chmod 的部分参数

参　　数	功　　能
-f,--silient,--quiet	以强制或安静方式工作，忽略大部分错误信息
-v	报告 chmod 对每个文件的处理信息
--reference=rfile	从指定文件 rfile 中获取权限值
-R,--recursive	以递归方式处理子目录及其中的文件

3．使用示例

```
#chmod a+x my_p1 my_p2          #为所有用户（all）对文件 my_p1、my_p2 增加执行权
#chmod u+rwx,go+rx myp          #为文件主增加所有权限，为组和其他人增加读和执行权
#chmod -R o-rwx /tmp/w123       #递归去掉/tmp/w123 内目录和文件对其他人的所有权限
#chmod -R 777 /tmp/*            #将/tmp 中的所有文件和各级子目录及内容权限设置为 777
```

5.2.3　改变文件的所有者（chown）

1．功能与用法

chown（change owner）的功能是改变文件的所有者，本质上是改变文件主的 uid。只有超级用户有权使用 chown。其用法为：

```
chown [ options ] owner[:group] file …
chown [ options ] :group file …
```

如果用户组被指定，则文件的所属组也将被修改。

2．参数说明

chown 的参数如表 5-5 所示。

表 5-5　chown 的部分参数

参　　数	功　　能
--dereference	改变符号链接最终对象的所有者，而非符号链接本身
-h,--no-dereference	与--dereference 相反，仅改变符号链接的所有者，而非链接对象
--from=cur_own:cur_grp	只改变与当前主和组匹配的目标。主和组中的一个可省略
-f,--silient,--quiet	以强制或安静方式工作，忽略大部分错误信息
-v	报告 chown 对每个文件的处理信息
--reference=rfile	从指定文件 rfile 中获取权限值
-R,--recursive	以递归方式处理子目录及其中的文件

3．使用示例

```
#chown root my_files              #将 my_files 的所有者改为 root
#chown zhang *.c                  #将当前目录下所有 C 语言文件的所有者改为 zhang
#chown -R gjshao /tmp/mydir       #递归将/tmp/mydir 的所有者改为 gjshao
#chown -R gavin:bin mydat         #递归将 mydat 的所有者改为 gavin，组改为 bin
#chown -R :bin mydat              #递归将 mydat 的组改为 bin
```

5.2.4　改变文件的组（chgrp）

chgrp（change group）的功能是改变文件所属组，其用法为：

```
chgrp [ options ] new_group file …
```

new_group 可以组名或组标识给出。chgrp 命令可由 chown 替代，其参数与 chown 相同。

5.2.5　ext2+文件系统的新增属性及其管理

1．ext2+的新增文件属性

从 ext2 文件系统开始，增加了很多新属性，新增的部分属性如表 5-6 所示。

表 5-6　ext2+增加的部分属性

属　　性	说　　明
A	文件被访问时不修改访问时间
a	只能以增加方式打开
c	自动压缩
D	目录改变时同步写入，类似在安装文件系统时使用了 dirsync 选项

（续表）

属　　性	说　　明
d	当使用 dump 备份时，不备份此文件
e	扩展格式（此属性不能去除）
i	不能修改，不能删除，也不能有链接指向此文件
s	安全删除。当文件被删除时，文件原来的内容将填充 0x00
S	文件修改时同步写入，类似在安装文件系统时使用了 sync 选项
u	当文件被删除时，内容将被保存，可以用于删除后的恢复

2．lsattr

lsattr 的功能是显示文件的 ext2+扩展属性，其用法为：

```
lsattr [ -RVdv ] file …
```

其参数如表 5-7 所示。

表 5-7　lsattr 命令的部分参数

属　　性	说　　明	属　　性	说　　明
-a	列所有文件属性，也包括“.”开始的文件	-R	递归处理
-d	列目录属性，而非其中文件属性	-v	显示文件版本号

3．chattr

chattr 的功能是对 ext2+的新增属性进行管理，其用法为：

```
chattr [ -RV ] [ -v version ] [ mode ] file …
```

权限 mode 的符号操作符为：+|-|=MODE，MODE 为表 5-6 所示的新属性，+用于增加属性，-用于去除属性，=设置属性时去除其他属性。

设当前目录内有文件 myfile，通过 ls -l x 和 lsattr x 得到它的权限和扩展属性分别为：

```
-rw-r--r--    1 root    root    3046   Oct 18 23:04   myfile
------------- myfile
```

使用 chattr +i myfile 为其增加 i 属性后，再使用 ls -l x 和 lsattr x 检查其权限和扩展属性为：

```
-rw-r--r--    1 root    root    3046   Oct 18 23:04 yfile
----i-------- myfile
```

可见 myfile 已经具有了 i 属性。使用命令

```
rm -f myfile
```

删除之，得以下错误输出：

```
rm: cannot remove 'myfile': Operation not permitted
```

这说明 i 属性对文件的删除起到了保护作用。如果使用命令

```
chattr -i myfile
```

去掉 i 属性后，就可以使用 rm 命令将其删除了。

5.3 文件系统管理

不同类型的操作系统总是按照自己的目的支持一些特定的文件系统。Linux 支持多种微机上常用的文件系统，Windows2000 及以后版本的 Windows 系统支持 FAT12、FAT16、FAT32 和 NTFS 等。关于 Linux 所支持的文件系统请阅读 filesystems 的在线手册，或在安装 kernel-doc 包后参阅/usr/share/doc/kernel-doc-*/Documentation/filesystems/内的相关*.txt 文档。

5.3.1 UNIX/Linux 支持的文件系统

1．msdos、umsdos、vfat、ntfs 文件系统

msdos 文件系统是早期 IBM 或 MSDOS 所支持的 FAT12 或 FAT16 文件系统。随着 MS Windows 的发展，在 DOS 文件的基础，又出现了 FAT32 文件系统。

umsdos 是 msdos 文件系统在 Linux 上的扩展，它在不牺牲 msdos 兼容性的前提下增加了长文件名、uid/gid、POSIX 文件权限和特别文件。

vfat 是以上 FAT12、FAT16、FAT32 等文件系统的总称。

ntfs 是 MS windows NT 及以后版本 Windows 系列可支持的文件系统。

2．minix 文件系统

minix 文件系统是 Linux 前身 minix 操作系统的文件系统。它有很多不足或限制：64MB 的分区限制；短文件名；只有一个时间日戳等。其依然存在主要是因为软盘（floppy）和内存文件（ramfs）系统。

3．ext、ext2、ext3、ext4

ext 是 minix 文件系统的扩展，现已经被 ext2 所替代，且已从版本 2.1.21 开始 Linux 在内核中取消了对它的支持。

ext2 是 Linux 系统高效、可靠的文件系统，可用在硬盘或可移动存储介质上。它是 ext 的扩展，被认为是基于速度和 CPU 使用最高效的文件系统。ext3、ext4 是 ext2 的相继扩展，它们在 ext2 的基础上增加了日志等功能。

4．iso9660 文件系统

iso9660 文件系统是符合 iso9660 标准的 CDROM 文件系统，它有两种标准：High Sierra 和 Rock Ridge。

High Sierra 是标准 iso9660 的先驱文件系统，在标准的 iso9660 文件系统内它能自动被识别。Rock Ridge 是一种 UNIX 格式的文件系统，在这种格式下可提供长文件名、uid/gid、POSIX 文件权限和设备特别文件等。

5．proc、sysfs

proc 是用于内核数据接口的伪文件系统，proc 不占用磁盘空间，在系统启动时自动产生并挂载到/proc，在系统关闭时自动消失。它可以理解为 PCB 表结构在内存中的登记，在 Linux 系统中可以理解为 task_struct 结构，其中记录了系统中每个进程的活动情况、资源占用等信息。当系统创建一个进程时，就要在/proc 中按进程的 pid 为进程创建一个目录，以记录该进程的动态行为和资源需求等。用户可以进入/proc 目录查看其中的内容。

对用户来说，proc 的大部分是只读的，通过它可得到系统参数，但是其中的某些文件是允许修改的，可通过修改其内容达到动态个性修改内核参数的目的。更多关于 proc 文件系统的信息可阅读 proc man 手册。

sysfs 是一种基于 RAM 的文件系统，它提供了一种向用户空间展现内核空间的对象、属性和链接工具。这里的内核空间对象是指 kobject 结构（这里不深入讨论），它是 Linux 在 2.6 中被引进的统一的设备管理模型，主要目的是使系统的所有设备在底层都具有统一接口，以便进行统一管理。sysfs 被安装在/sys 目录下。在/sys 下有 block、bus、dev、fs、power 等目录，分别对应相应类型的设备，在它们的每个下面还有各种不同子类型，用户可以使用 tree 命令查看其中的层次结构。udev 可使用/sys 的信息动态管理/dev 下的设备文件，也就是说是 sysfs 动态设备管理的基础。

6．sysv、s5、xenix

s5 是 UNIX 系统的文件系统。xenix 操作系统是 UNIX 在微机系统上的一种实现，它所使用的文件系统是 xenix。sysv 是 UNIX sysv 文件系统在 Linux 系统的一种实现。

7．smbfs、cifs

smb 是用于 Windows for Workgroup、Windows NT 和 Lan Manger 支持 smb 协议的网络文件系统。在 Linux 系统上可以通过 samba 等系统实现与 Windows 系统的共享。smbfs 类型为 smb 或 cifs。

8．nfs

nfs 是一种网络文件系统，用于在 UNIX 系统间通过网络进行文件共享，用户可以把 nfs 服务器提供的共享目录挂载到本地目录下，然后可以像对本地文件系统一样操作 nfs 系统的内容。

9．jfs

jfs（jounaling file system）是日志文件系统。提供日志功能的文件系统是当今的发展趋势。当今主流公司的小型机以上操作系统的文件系统大多采用或支持 jfs 文件系统。

采用 jfs 的操作系统，在将数据写入外存前先写日志文件，以便在外存操作出故障后可从日志文件恢复，增加了系统的安全性。ext3+都是日志文件系统。

5.3.2 UNIX/Linux 系统使用的存储设备

Linux 系统使用的存储设备或介质有软盘、硬盘、光盘、U 盘、磁带和多种存储卡等。关于设备文件及设备文件的描述请在安装内核源代码包后，可参见内核源代码包中的 Documentation/devices.txt 文件（参见§7.7）。

1．IDE 硬盘存储设备

一般情况下，在一个计算机内最多可以安装 4 个 IDE 硬盘。它们可分别是主（master）驱动器上的主（master）硬盘、从（slave）硬盘和从（slave）驱动器上的主、从硬盘。

IDE 硬盘设备的形式为/dev/hdmn，其中，m∈(a,b,c,d,…)代表驱动器号，/dev/hda，/dev/hdb 为主控制器上的主硬盘和从硬盘，/dev/hdc，/dev/hdd 代表从控制器上的主、从硬盘；n∈(1,2,3,…)代表第 m 个硬盘上的分区号，如/dev/hda 代表第一个硬盘，/dev/hda1、/dev/hda2、/dev/hda3、/dev/hda4 分别是/dev/hda 的 4 个基本分区，/dev/hda5、/dev/hda6 则为/dev/hda 上的第一和第二个逻辑分区。

2．SCSI 存储设备

SCSI（Small Computer System Interface）接口是小型计算机接口的简称，是一个多用途的 I/O 接口，除可接磁盘外，还可接磁带机、光驱，扫描仪等设备。SCSI 总线上能连接的设备类型很多，结构和控制上也与 IDE 不同，这里不做过多介绍。

SCSI 存储设备的形式为/dev/sdmn，其中，m∈(a,b,c,d,…)为字符代表物理设备，n∈(1,2,3,…)为物理设备 n 上的逻辑分区编号。

SCSI 上硬盘设备的命名方法与 IDE 硬盘设备的命名方法相同。例如，/dev/sda、/dev/sdb 分别为第 1、2 个 SCSI 硬盘，/dev/sda1、/dev/sda2 分别代表第 1 个 SCSI 硬盘的第 1、2 分区。

3．USB 存储设备

Linux 以 SCSI 方式支持 USB 存储设备。一般来说，USB 存储设备是系统中正在使用的所有 USB 设备最后一个的下一个。

如果系统中没有使用 SCSI 存储设备，则新来的 USB 设备可根据所使用的 USB 物理设备不同而可能使用/dev/sda、/dev/sda1、/dev/sda2，…或/dev/sdb、/dev/sdb1 等。如果系统使用了 SCSI 硬盘，且所使用的最后一个 SCSI 硬盘设备为/dev/sdb，则新来的 USB 等移动设备可用的第一个 SCSI 设备为/dev/sdc，相应地，其上若有分区，则分别为/dev/sdc1、/dev/sdc2…

USB 等移动存储设备的类型和用法复杂，有有分区的，有无分区的。比如，软盘、手机等存储上无分区，而移动硬盘上面有分区，且可能不止一个，应具体设备具体对待、具体处理。

4．光驱和刻录机

一般来说，光盘驱动器为/dev/cdrom，该设备是某个硬盘设备的符号链接或硬链接。例如，在 Linux 系统中光盘设备/dev/cdrom 是/dev/hdc 的一个符号链接，而在现在的系统中/dev/cdrom 大多是/dev/sr0 的一个链接。

在 Linux 系统中，默认刻录机设备是/dev/cdwriter，它被链接到/dev/sr0 或/dev/sg0 上。

用户可能直接使用被链接的设备/dev/sr0 或/dev/sg0 作为光驱或刻录设备。

5．软盘设备

早期大多数操作系统都支持软盘，尽管现在软盘已经被淘汰，但在构造和使用文件系统映像时还要用到它。

软盘设备名为/dev/fdnts，其中 fd 不变，为软盘标识；n∈(0,1,…)为软驱编号；t∈(d,h,u)为软驱类型，d 和 h 用于 5”盘，分别表示双面或高密，u 用于 3”盘，表示以双面和高密。s∈(360，720，1200，1440，2880，…)表示软盘的容量。

Linux 系统常用软盘设备为/dev/fd0（第一软驱），默认容量为 1.44MB。

6．磁带设备

磁带机是 UNIX/Linux 系统用来数据备份、系统安装等操作的常用设备。在 Linux 系统中支持多种磁带设备，IDE 的磁带设备文件为/dev/ht0、/dev/ht1 等，SCSI 的磁带设备为/dev/st0、/dev/st1 等。

5.3.3 磁盘分区管理与文件系统的创建

文件系统是建立在存储介质上的，对存储介质的使用要首先进行规划，如对硬盘的使用必须先在其上创建（物理或逻辑分区）分区等。其次是对分区进行格式化，在其上创建文件系统，然后才能使用。很多时候，用户使用设备时并没有做这些工作，那是因为在这之前已经做过了。

对磁盘分区的规划与管理需要很多实践经验，若操作不慎可能会造成硬盘数据丢失，初学者可以在试验用机上进行操作。关于磁盘分区结构信息可参见第 7 章。

1．使用 fdisk 和 parted 进行分区管理

用于管理硬盘上分区的命令有 fdisk 和 parted，在图形界面下还有 gparted 等，这里只简单介绍 fdisk 和 parted，主要目的是用于对磁盘分区的显示与查询。对于 GPT 格式磁盘，还有 gdisk 等。

1）使用 fdisk 进行分区管理

（1）fdisk 命令的功能及用法。

fdisk 功能是在硬盘上创建、删除或修改分区。其用法为：

```
fdisk [-u] [-b sectorsize] [-C cyls] [-H heads] [-S sects] device
fdisk -l [-u] [device …]
fdisk -s partition …
```

（2）参数说明。

fdisk 的选项如表 5-8 所示。

表 5-8　fdisk 的选项

参　　数	功　　能
-b sectorsize	指定磁盘扇区的大小。通常为 512B，1024B 或 2048B

（续表）

参　　数	功　　能
-C cyls	指定磁盘柱面数
-H heads	指定磁盘的磁头数。通常为 255 或 16 等
-S sects	指定磁盘每道上的扇区数。通常为 63
-l	以柱面方式列指定盘上的分区。当给出-u 参数时以扇区为单位列分区
-s partition	显示指定分区的大小
-u	以块为单位显示分区的大小

当以

fdisk <硬盘设备名>

方式执行时，fdisk 进入交互方式，用户可以在交互方式下使用表 5-9 中的命令对指定磁盘进行操作。

Command (m for help):

当用户输入 m 后，显示一个帮助菜单。

表 5-9　fdisk 的交互方式常用命令表

参　　数	功　　能	参　　数	功　　能
a	设置活动分区	p	列出硬盘分区表
d	删除指定分区	q	退出，但不保存以前的操作
l	列出所有支持的分区类型	t	更改分区类型
m	列出帮助命令菜单	u	切换显示分区的单位
n	创建一个分区	v	校验分区表
o	创建一个空 DOS 分区	w	写分区表，并退出

（3）使用 fdisk 分区示例。

```
#fdisk                    #fdisk 命令用法信息
#fdisk -l                 #显示所有硬盘或 U 盘的分区信息
#fdisk -l /dev/sda        #显示硬盘/dev/sda 的分区信息
```

2）使用 parted 进行分区管理

（1）功能及用法。

parted 的功能也是磁盘分区管理，其用法为：

parted [options] [device [command [arguments]]]

常用参数有-l：列所有块设备分区表。-h：帮助。-v：显示版本号。device 为所要操作的存储设备，如/dev/hda 或/dev/hdb 等。command 为对设备操作的子命令。arguments 为子命令所需的参数。

（2）parted 的命令及示例。

当不带任何选项及参数运行 parted 时会进入交互方式。在交互方式下，可以使用的

command、参数及示例如表 5-10 所示。

表 5-10　parted 的命令

命　　令	参　　数	示　　例	说　　明
quit	无	quit	退出 parted
help	帮助内容	help cp	对命令 cp 进行帮助
print	[MINOR]	print	显示当前盘全部或分区 1 的信息
check	MINOR	check 1	检查 1 号分区的文件系统
cp	[DEVICE] FROM-MINOR TO-MINOR	cp /dev/sdb 2 3	复制 sdb 的分区 2 到当前设备的 3 号分区
mklabel	LABEL-TYPE	mklabel msdos	创建一个 msdos 标签
mkfs	MINOR FS-TYPE	mkfs 2 fat32	在 2 号分区上创建 FAT32 文件系统
mkpart	PART-TYPE [FS-TYPE] START END	mkpart primary 0.0 800.0	创建一个基本分区，从 0～800MB，类型默认
move	MINOR START [END]	move 2 150	将 2 号分区移到 150MB 开始处
name	MINOR NAME	name 2 'myDsk'	将分区 2 命名为 myDsk
rescue	START END	rescue	交互式恢复一个分区
resize	MINOR START END	resize 3 200 850	将分区 3 调整 200～850MB
rm	MINOR	rm 3	删除 3 号分区
select	DEVICE	select /dev/sdb	选择/dev/sdb
set	MINOR FLAG STATE	set 1 boot on	将分区 1 设为 boot 且激活它

说明：表中的 MINOR 为分区号；LABEL-TYPE 为 msdos、sun、bsd 或 mac 等；FS-TYPE 为文件系统的类型，常见类型有 ext2、ext3、ext4、fat16、fat32 和 linux-swap 等；PART-TYPE 为 primary（基本的）、extended（扩展的）或 logical（逻辑的）。

（3）命令行方式。

表 5-10 中的命令也可以直接写在 parted 的命令行上，让 parted 以非交互式执行，例如：

```
#parted help                 #获得帮助
#parted /dev/sda print       #显示/dev/sda 的分区情况
#parted set 1 boot on        #将分区 1 设为 boot 且激活它
```

2．创建文件系统（mkfs）

UNIX/Linux 系统对存储介质的使用是要先进行格式化的，除非该介质是已经格式化过的。大部分 UNIX/Linux 系统对磁盘介质的格式化过程包含在文件系统的创建过程中。创建文件系统的命令是 mkfs。

1）功能及用法

mkfs 是创建文件系统的总控程序，它会根据将要创建的文件系统类型去调用相应的创建程序来完成文件系统的创建工作，可以用于创建文件系统的子命令如表 5-11 所示。

表 5-11　用于创建文件系统的命令

命令格式 1	命令格式 2	功能描述
mkfs.ext2，mkfs.ext3，mkfs.ext4，mkfs.ext4dev	mke2fs	创建 ext2+文件系统
mkfs.msdos，mkfs.vfat	mkdosfs，mkfsfat	创建 MSDOS 文件系统
mkfs.ntfs	mkntfs	创建 NTFS 文件系统
mkfs.minix		创建 minix 文件系统
	mkisofs	创建 iso9660 光盘文件系统
	mkswap	创建 swap 文件系统

说明：mkfs.*文件的多少与系统支持的文件系统有关，表中只是常用者。

mkfs 的用法为：

```
mkfs   [ -v ] [ -t fstype ] [fs-options ] filesys [ blocks ]
```

2）参数及说明

mkfs 命令的参数如表 5-12 所示。

表 5-12　mkfs 的部分参数

参　　数	功　　能
blocks	文件系统的总块数，一般不强行指定，由系统默认
filesys	文件系统。一般为设备文件，也可为普通文件。若为普通文件，则必须指定大小
-t fstype	文件类型，若不指定则使用默认文件系统
-v	显示冗余信息
-c	创建文件系统前做坏块检查。此选项在某些文件系统中可能不被支持
-l filename	从文件中读取坏块信息。此选项在某些文件系统中可能不被支持

3）文件系统创建示例

```
#mkfs -t vfat /dev/sdb          #在移动磁盘上创建 vfat 文件系统
#mkfs -t ext2 /dev/sdb          #在移动磁盘上创建 ext2 文件系统
#mkfs -t ntfs /dev/hd10         #在 IDE 硬盘的/dev/hda10 分区上创建 NTFS 分区
```

3．创建光盘文件系统（mkisofs/genisoimage）

1）功能及用法

传统的创建光盘文件系统的命令是 mkisofs，在现代 Linux 系统中它有一个替代命令 genisoimage，这里将它们理解为同一命令。光盘文件系统有多种格式，利用 Linux 系统提供的光盘文件系统创建命令 mkisofs/genisoimage 可以创建包括具有 Rock Ridge 属性在内的多种 iso9660 文件系统。一般不使用它们直接刻录光盘，而常用它创建一个光盘映像文件。它们的用法为：

```
mkisofs   [ options ] [ -o filename ] path [ path … ]
genisoimage   [ options ] [ -o filename ] path [ path … ]
```

2）参数说明

命令 mkisofs/genisoimage 常用参数如下。

path：欲刻录的文件名、目录名或目录树名；-o filename：光盘映像文件名；-b myboot_img：用于指定引导盘映像的文件名为 myboot_img（说明：引导盘映像必须以相对路径的方式存在于 path 中，否则刻成的光盘映像不能引导）；-r/-R：刻录具有 Rock Ridge 属性的光盘映像，也就是 UNIX/Linux 格式的光盘映像（若不使用-r/-R，则文件名将被截成 8.3 格式的 MSDOS 格式；-r 和-R 的主要区别是，-R 保持文件的 uid 和 gid，而-r 将所有文件的 uid 和 gid 均置为 0）。

该命令还有很多参数可供选择，这里不再做具体介绍，请参阅相关手册。

3）应用举例

（1）创建 iso 文件系统。

```
#将目录 cd_dir1、cd_dir2 等的内容“刻录”到映像文件 mycd.iso
#mkisofs -o mycd.iso cd_dir1 cd_dir2 …
```

注意：因未指定 Rock Ridge 格式，则 cd_di1、cd_dir2 中的长文件名将被截为 8.3 格式。

（2）创建具有 Rock Ridge 属性的文件系统。

```
#genisoimage -o myRcd.iso -R my_dir
```

（3）创建可启动的光盘。

```
#mkisofs -o my_bootcd -b myboot_img my_cd_dir
```

说明：-b 用于指定映像的文件名为 myboot_img，引导盘映像必以相对路径的方式存在于 my_cd_dir 中。

4．光盘刻录（cdrecord/wodim）

由于 mkisofs 不直接在光盘上构造文件系统，Linux 还提供有命令 cdrecord 和 wodim 用于将光盘映像或文件刻录到光盘上。其用法为：

```
cdrecord [ general options ] dev=device [ track options ] track1 … trackn
wodim [ general options ] dev=device [ track options ] track1 … trackn
```

cdrecord 是一个复杂的系统，涉及很多 iso9660 文件系统和刻录机相关技术参数，还要和不同厂家的刻录设备打交道。

dev=device 用于指定刻录设备，其中的 device 就是系统内安装的刻录机设备。但由于不同厂家的刻录设备不同，不同系统内的刻录设备也不完全一致，不好确定，不过这不是问题，因为 cdrecord 支持设备自动扫描功能，也就是说，当你不使用 dev=device 指定刻录设备时，cdrecord 将为你自动搜索并设定刻录设备。对刻录设备进行扫描的方法是：

```
#wodim --devices
#wodim -scanbus
```

这种做法是可行的，因为一个系统中往往只有一台刻录机，但 wodim 的 man 手册也会告诉用户，这种搜索有时可能是不可靠的。

在 Linux 中刻录设备为/dev/cdwriter 或/dev/sg0 或/dev/sg1 等，操作者可以直接使用文件名。对于 SCSI 刻录设备，设备的命名形式为 scsibus，target，lun，其中 scsibus、target 和 lun

分别为刻录设备所连接的总线、目标和设备号。可以通过 dev=help 或-scanbus 参数来查询可用设备。

刻录光盘的示例如下：

```
#wodim -scanbus                                  #扫描 SCSI 设备
#cdrecord dev=/dev/cdwriter myiso.img            #将光盘映像 myiso.img 刻录到光盘
#mkisofs -R my_dir | wodim dev=/dev/cdwriter     #通过管道将文件直接刻录到光盘
#mkisofs -R my_dir | cdrecord dev=/dev/sg0       #同上
#cdrecord dev=help                               #查询刻录设备列表及简单描述信息
#wodim -v -eject speed=4 dev=0,3,0 myiso.iso     #将光盘映像文件刻录到光盘，完毕后打开光驱
```

5.3.4　文件系统的使用

UNIX/Linux 系统可以使用自己支持的文件系统。欲使用已经存在的文件系统，必须首先使用安装命令 mount 进行安装，然后才能使用，使用完毕后要用拆卸命令 umount 进行拆卸。

当一个外来文件系统或设备被安装在某一个目录上后，该文件系统或设备上的内容就可通过该目录在系统中得到反映，这个被安装的文件系统就可以作为整个文件系统的一部分来使用了，从而对该目录的存取变成了对设备的存取。

为了保证对存储介质的正确使用，必须按“安装→使用→拆卸”的顺序来进行操作。如果不按规定操作而中途把介质强行取出，则可能造成存储介质上文件系统的损坏、数据不完整或丢失。

受损的文件系统必须经过（使用 fsck）清理或修复后才能再继续使用。

在 UNIX/Linux 系统中，所有可以访问的文件或文件系统都必须在文件系统的目录树上。mount 的作用就是要把单独、分离的文件系统挂载在 UNIX/Linux 文件系统的某个目录上。理论上，外来文件系统可以安装在文件系统的任何目录上。在早期的系统中，要求安装位置或称为安装点必须是一个空目录。在现在的系统中，安装点也可以是非空目录，但有一个限制，当在某一个非空目录上安装外来文件系统后，该目录中原来的内容将暂时无法“看到”了，直到安装的部分被拆卸掉。为了解决安装位置的问题，UNIX/Linux 系统提供了一个可以被广泛使用的安装点/mnt。在现在的版本中，往往还在/mnt 目录下开辟有一些子目录，如/mnt/floppy、/mnt/cdrom 等用于不同目的的安装。必要时，用户也可以在/mnt 下创建自己的目录，如/mnt/hd、/mnt/usb 等用于安装硬盘分区和 U 盘等。在现在的 Linux 系统中，图形界面自动安装时所使用的安装点为/media/或/var/run/目录。

1．文件系统安装（mount）

1）功能及用法

mount 命令的功能是安装文件系统。其用法为：

```
mount [-lhv]
mount -a [-fFnrsvw] [-t fstyp] [-O optlist]
mount [-fnrsvw] [-o opts [, …]] dev | dir
mount [-fFnrsvw] [-t fstyp] [-o opts] dev dir
```

这里的 dev 为设备或文件系统，dir 为安装目录或安装点，-o 选项如表 5-13 和表 5-15 所示。

方式 1 用于显示系统中已经安装的所有文件系统；方式 2 用于根据/etc/fstab 自动安装；方式 3 用于根据/etc/fstab 的设定，将指定设备 dev 安装到指定位置 dir，设备或位置只选其一；方式 4 与方式 3 意义相同，但设备或位置必须同时具备。

与 mount 命令相关的文件有/etc/mtab、/proc/mounts、/proc/PID/mounts、/etc/fstab。/etc/mtab 与/proc/mounts 的内容是系统现行安装的文件系统表。/etc/fstab 是 UNIX/Linux 系统实现文件系统自动挂载的配置文件（见后述）。

2）参数说明

mount 命令常用的参数如表 5-13 所示。

表 5-13　mount 命令常用参数

参　　数	功　　能
-a	安装所有由/etc/fstab 管理的文件系统
-t fstyp	指定文件类型，若不指定则系统将使用-t auto 自行测试
-r	以只读方式安装
-o	用于设置安装选项（详见表 5-15）。使用-o loop 参数可将一个映像文件上的文件系统安装在系统上

3）mount 命令使用示例

```
#mount                              #列系统内安装的设备或文件系统
#mount -t vfat /dev/sda1 /mnt/usb   #将 U 盘上 vfat 格式的系统安装在/mnt/usb 上
#mount -t vfat /dev/sdc /mnt/fd     #将 U 盘上 vfat 格式的系统安装在/mnt/fd 上
#mount -r -t ext2 /dev/fd0 /mnt/fd  #将软盘上 ext2 格式的文件系统以只读方式安装
#mount /dev/cdrom /mnt/cdrom        #将光盘安装在目录/mnt/crcom 上
```

2．文件系统卸载（umount）

当一个被安装的文件系统使用完毕后，应该将其拆卸下来，以确保文件系统的完整性。拆卸已经安装文件系统的任务由 umount 完成。若是软盘、U 盘和移动硬盘等移动设备，在取走设备前必须拆卸。系统设备也可不拆卸，待关闭系统时由系统自动完成拆卸工作。

umount 的简易用法为：

```
umount dev_mounted          或
umount dir_mounted
```

设设备/dev/sdb1 上的文件系统被安装在/mnt/usb 上，则可用两种方法拆卸它：

```
#umount /dev/sdb1           或
#umount /mnt/usb
```

与 umount 具有相同功能的命令是 eject，但它专用于拆卸诸如光盘和 U 盘等移动设备。

若有用户正在已安装的目录工作，拆卸时会出现“device is busy”信息。解决问题的办法是让工作者离开此目录，然后再拆卸。

当光盘被成功安装后，直到被拆卸后，光驱将无法再打开，若要打开光驱，则必须首先卸载光驱设备。最简单的拆卸命令是：

```
#eject
```

3．映像文件的挂载

安装设备上的文件系统是和某个设备密切相关的，安装一个文件系统的映像文件使用什么设备呢？系统提供的 loop 设备可用于此目的。将映像文件挂载或安装到文件系统时需要使用-o 参数和 loop 设备。方法是：

```
mount -t fstype -o loop[=loop_dev] fs_img mnt_dir
```

这里的 loop_dev 为/dev/目录下的文件（loop0~loop7 或更多）。安装时，可只使用 loop 参数而不指定 loop 设备，由 mount 自己查找一个可供使用的空闲设备。映像文件挂载示例如下。

（1）将软盘映像文件 my_dos 上的 DOS 文件系统安装到/mnt/fd。

```
#mount -t vfat -o loop my_dos /mnt/fd
```

（2）将光盘映像文件/tmp/myiso 的内容复制到当前目录。

```
#mount -t iso9660 -o loop /tmp/myiso /mnt/cdrom        #安装在目录/mnt/cdrom
#cp -r /mnt/cdrom/*    .                               #复制其内容到当前目录
#umount /mnt/cdrom                                     #拆卸映像文件
```

4．文件系统的自动挂载

1）功能及配置文件

手工挂载的文件系统可以手工拆卸，也可在系统关闭时由系统自动拆卸。UNIX/Linux 系统可以自动挂载设备，这表现在当系统启动时一些设备被自动挂载，也表现在某些设备连接到计算机系统时，例如，光盘被放入光驱并关上仓门后，系统会将其自动挂载到指定位置。

控制设备自动挂载的配置文件为/etc/fstab，按传统做法，能够自动挂载的设备或文件系统都必须在此文件中定义。/etc/fstab 的结构为：

```
<filesystem> <mount_point> <fstype> <options> <dump> <pass>
```

其中每个域的意义如表 5-14 所示。

表 5-14　/etc/fstab 文件说明

域	说　　明
Filesystem	要挂载的文件系统或设备
mount_point	安装位置
fstype	文件系统类型
options	安装选项。可同时使用多个选项，选项间用逗号分隔。通用部分选项参见表 5-15
dump	备份标志：0—不备份，1—备份。Dump 使用
pass	开机时系统自动检查的顺序，fsck 使用。0—不检查，1—检查，2—其他

表 5-15　/etc/fstab 文件挂载部分选项

选　　项	说　　明
defaults	默认值。选项包括 rw、suid、dev、exec、auto、nouser 和 async
async	以异步方式处理文件系统的 I/O

（续表）

选　项	说　明
Sync	以同步方式处理文件系统的 I/O
auto	可使用-a 自动安装，开机时也会自动安装
noauto	非 auto。使用-a 时不能自动安装，开机时也不自动安装
dev	可解读文件系统上的字符或块设备
nodev	不可解读文件系统上的字符或块设备
exec	可执行文件系统的二进制文件
noexec	不可执行文件系统的二进制文件
suid	启用 suid 和 sgid 属性
nosuid	禁用 suid 和 sgid 属性
user	允许用户 user 挂载此文件系统，系统默认时只让 root 使用
nouser	禁止一般用户挂载此文件系统
users	使一般用户可挂载此文件系统，用于桌面环境
rw	以读写方式挂载
ro	以只读方式挂载
atime	存取文件时，更新 i 节点存取时间。此为默认动作
noatime	存取文件时，不更新 i 节点存取时间

某系统的/etc/fstab 如下所示。

```
#/etc/fstab: static file system information.
#<file_system>   <mount_point>   <fstype>   <options>        <dump>   <pass>
proc                                    /proc          proc    nodev,noexec,nosuid   0   0
UUID=810cb0fb-3bc1-4782-82a1-a1a239110504 /            ext4    errors=remount-ro     0   1
UUID=01a2be0e-9e3f-4755-88e8-d1fde3250f69 none         swap    sw                    0   0
/dev/fd0                                /media/floppy  auto    rw,user,noauto,exec   0   0
/dev/cdrom                              /media/cdrom   udf,iso9660   noauto,owner,ro 0   0
```

其中的 UUID 所对应的设备可用命令 blkid 来查询。

对于已经在 fstab 内定义的文件系统可以使用不带安装点的 mount 命令安装，例如，在 fstab 中定义了/dev/cdrom 和/dev/fd0 的安装位置分别为/media/cdrom 和/mnt/floppy，则使用命令

```
#mount /dev/fd0        和
#mount /dev/cdrom
```

可将光盘和软盘分别安装在目录/mnt/cdrom 和/mnt/floppy。

需要说明的是，在现在的 Linux 中，所有的移动设备都无须在/etc/fstab 中定义，但仍可被自动安装，安装位置会因系统不同而不同，一般是/media 或/run/media。

2）文件系统自动挂载实例

对于多个操作系统共享硬盘的计算机系统，在 Linux 系统中可以通过安装的办法访问其他分区上的文件系统。设 Windows 系统的 D:盘所对应的设备为/dev/hda5，其上的文件系统为 fat 格式，则可通过命令

```
#mount /dev/hda5 /mnt
```

将其安装在/mnt 目录。也可以通过设置开机自动安装的办法将其安装在/mnt/wind 目录，方法如下：

（1）创建一个安装点/mnt/wind。

```
#mkdir -p /mnt/wind                    #只需创建 1 次，若已经存在，则不需再创建
```

（2）编辑/etc/fstab 文件，在其中增加一行，其内容为：

```
/dev/hda5 /mnt/wind vfat rw    0    0
```

配置完成后，每当系统重新启动后，Windows 的 D 盘就会自动安装在/mnt/winD 上。

5．查找当前安装的文件系统（findmnt）

findmnt 命令的作用是查找所有已经安装的文件系统或搜索文件系统，其用法为：

```
findmnt [options]
findmnt [options] device|mountpoint
findmnt [options] [--source] device [--target|--mountpoint] mountpoint
```

findmnt 可以扫描/etc/fstab、/etc/mtab 和/proc/self/mountinfo。device 为用于指定源设备，可以是设备名，主、次设备号，LABEL=label，UUID=uuid，PARTLABEL=label 和 PARTUUID=uuid。

如果 device 或 mountpoint 没有被指定，则将显示所有的文件系统。

默认情况下，以树状方式显示，选项-l 或--list 让其以列表方式输出。常用的选项有--D/-df：产生 df 命令输出格式的输出。-r/--raw：原始方式输出。-s/--fstab：输出/etc/fstab。-t/--types list：指定文件系统类型。-T/--target path：指定挂载路径。-M/--mountpoint path：指定安装路径。-m/--mtab：搜索/etc/mtab。-k/--kernel：指定搜索/proc/self/mountinfo。

findmnt 的应用示例如下。

```
#findmnt                      #树状方式列出所有已经挂载的文件系统
#findmnt -l                   #表状方式列出所有已经挂载的文件系统
#findmnt --df                 #产生 df 命令格式的输出
#findmnt --raw                #原始方式输出
#findmnt -t ext4,iso9660      #输出所有已经挂载 ext4 和 iso9660 文件系统
#findmnt --fstab -t ext4      #输出/etc/fstab 中定义的已经挂载的 ext4 文件系统
#findmnt | grep /dev/sr0      #检查光盘文件系统是否被安装。同 findmnt -S /dev/sr0
```

6．以裸方式使用文件系统或设备

1）裸设备的概念

裸设备（Raw Device）是指没有文件系统的设备，例如，一个软盘、U 盘、硬盘或一个硬盘分区等都可被看成一个裸设备。裸设备也叫原始设备。

以裸或原始方式使用设备就是将设备直接作为文件来使用。因为在 UNIX/Linux 系统中设备是作为文件来管理的，所以设备可直接作为文件来使用。除非用户目的特别明确，设备直接作为文件使用是危险的，因为系统把设备直接作为文件来使用时，会忽略其上的文件系

统。若向其上写信息，则其上的原有文件系统将被覆盖，也就是说原来文件系统及其内容都将擦除并重写。

以原始方式使用设备或文件系统常用于构造设备映像文件，也可以进行介质复制或克隆。当然，若要对一个硬盘或分区进行重新规划或格式化，则必须以裸方式使用设备。欲将以裸方式使用写过的设备再以文件系统方式使用，还必须重新在其上构造文件系统。

2）裸设备使用举例

映像文件是指光盘、磁盘或其他存储的文件系统及内容在 UNIX/Linux 或 Windows 系统上的硬拷贝。

构造映像文件的方法是通过命令 cp 或 dd 将含有某个文件系统的存储介质（如光盘、软盘、U 盘、硬盘分区）上整体或部分内容以文件的形式完整地复制到指定文件中。映像文件的使用示例是制作映像文件或介质复制，示例如下。

```
#cp /dev/fd0 /tmp/fd_img              #制作软盘映像文件
#cp /dev/cdrom /tmp/cdimg.iso         #制作光盘映像文件，可能需要更大的存储空间
#cp /dev/sdb /tmp/u_img               #制作 U 盘映像文件，可能需要更大的存储空间
#dd if=/dev/cdrom of=cd.iso           #dd 比 cp 效率要高（参见 5.4.3 节）
##沿相反方向也可把映像文件复制到存储介质上，以达到复制介质的目的
```

5.3.5　文件系统的检查、修复与同步

几乎在所有的 UNIX/Linux 系统中，受损的文件系统都是不能被正常安装的，必须经检查和修复后才能被正常使用。文件系统修复工具为 fsck。

文件系统的检查有时是系统自动进行的，比如，Linux 在文件系统被安装一定次数或使用一定时间后要对其进行检查或清理。Windows 系统也经常在启动时做文件系统的检查工作。

1．文件系统要求清理的原因

有很多原因会引起文件系统的损坏或不一致，如意外掉电、非法关机、强行拔出未拆卸的设备等。

文件系统受损或不一致表现在：文件真实大小与分配的块不一致；文件的链接数错误；i 节点内容错；丢失的块与丢失的文件；空闲块总数或空闲 i 节点总数错等。检查文件系统的目的就是要修复以上问题。

2．fsck

1）功能与用法

fsck 的功能是检查、修复文件系统和报告文件系统情况。其用法为：

```
fsck [ opts ] [ -t fstyp ] [ filesystems ]
```

fsck 一般不用于对已经安装且正在使用的文件系统进行检查。在检查过程中发现的丢失（孤儿）文件将被拯救。拯救后的文件以 i 节点号为文件名，存放在文件系统的目录/lost+found 中。根据所用参数的不同，在 fsck 工作过程中可能会显示很多信息。

像 mkfs 一样，fsck 也是一个前端总控程序，在执行过程中将根据不同的文件系统而调用

相应文件系统的检查程序。相应文件系统的检查程序以 fsck.*的形式存放在系统目录/sbin 下。当不带参数运行 fsck 时，fsck 将根据/etc/fstab 的内容按顺序进行操作。

2）参数及说明

fsck 的常用参数如表 5-16 所示。

表 5-16 fsck 的常用参数

参　　数	功　　能
filesystem	系统支持的文件系统或设备，如/dev/fd0、/dev/hda1
-A	按/etc/fstab 顺序检查文件系统，一般在系统初始化时自动进行
-a	检查过程中自动修复遇到的错误
-C	检查时显示进度条
-N	不进行真正操作，只显示真正操作时应采取的动作
-R	当指定-A 时，跳过根文件系统
-r	执行被检查文件系统的统计信息
-t fstyp	指定文件系统类型
-y	所有回答均默认为 yes（与-a 互斥）

3）示例

（1）系统开机问题检查。

当系统遇到非法关机等情况时，在下次开机过程中可能要对文件系统进行检查，且可能会遇到自动检查通不过的情况，此时系统不再继续启动，而是直接出现提示符#进入单用户模式，或者出现：

```
Give root password for maintenance
 (or press Control-D to continue):
```

提示，要求用户输入 root 密码进入系统维护模式或^D 继续启动。此时用户能做的是输入 root 密码行系统维护模式，然后手工处理启动时遇到的问题：手工清理文件系统后重新启动。

设系统的引导分区和根分区分别为/dev/sda2 和/dev/sda10（可参见/etc/fstab），检查方法如下所示。

```
#fsck   -a                    #检查所有文件系统或依次检查每个文件系统
#fsck   -y /dev/sda2          #检查/dev/sda2
#fsck   -y /dev/sda10         #检查/dev/sda10
#reboot                       #重新启动
```

（2）检查软盘上的文件系统。

```
#fsck -t ext3 /dev/fd0        #设软件设备为/dev/fd0，其上的文件系统为 ext3
#fsck -y -t ext2 /dev/sdb     #设使用的设备为 USB 软驱，如/dev/sdb
```

3．sync

sync 的功能是同步文件系统，将所有缓冲区的内容写到存储设备，并更新超级块。其用法为：

sync

如果计算机系统的用电环境是安全的，并按规定进行操作，文件系统一般是不会受损的，因此不需要人为地进行同步，必要时系统会自动做此项工作，但在即将出现停电且又不能马上关机等特殊情况下，为了将文件系统所受损失降到最小，可用 sync 进行同步，把各种 I/O 缓冲区的内容写到存储设备上。

5.4　与文件系统管理相关的其他命令

5.4.1　文件综合查找命令（find）

1．功能及用法

find 命令用于文件的查找，从而定位文件在指定文件系统中的位置，其用法为：

find [path ...] [expression] ...

find 在文件查找过程中有很多指标（参见表 5-17）供使用。在 find 的所有参数中，位于命令名后第一个选项前的参数为查找位置，若无则默认为当前目录。如果也没有指定参数或查找指标，则默认为-print。

2．参数描述

find 功能强大，有很多用于搜索和查找的指标，这些指标就是用法中的 expression。expression 由操作符连接参数、判断和动作组成。组成 expression 的部分参数如表 5-17 所示。

表 5-17　find 命令的部分参数

参数类型	参　　数	意　　义
选项	-daystart	从当日开始计算时间，否则从 24 小时前。配合时间参数使用
	-depth	在查找完当前目录后再进入子目录
	-follow	跟踪符号链接
	-mount/-xdev	不搜索在其他文件系统上的内容
测试	数字	n 表示=n；-n 表示<n；+n 表示大于 n
	-name pattern	指定文件名。pattern 中可以包含通配符
	-amin n	指定最近曾访问时间，单位为分钟。参见“数字”
	-cmin n	指定最近状态被修改时间。参见-amin
	-mmin n	指定最近曾修改时间。参见-amin
	-atime n	指定最近曾访问时间，单位为 24H。参见-amin
	-ctime n	指定最近状态被修改时间。参见-atime
	-mtime n	指定最近被修改时间。参见-atime
	-anewer file	存取时间较 file 近
	-cnewer file	状态改变时间较 file 近

（续表）

参数类型	参　　数	意　　义
测试	-newer file	修改时间较 file 近
	-empty	空文件或空目录
	-executable/-readable/-writable	可执行/可读/可写
	-fstype type	指定文件类型
	-group gname	指定组名或 gid
	-user uname	指定用户名或 uid
	-perm PERM	指定权限。PERM 可为+nnnn、-nnnn 或 nnnn
	-regex pattn	指定正则表达式
	-size n[cwbk]	指定大小，默认为块（b）
	-type t/-xtype t	指定文件类型：b，c，d，p，f，l 和 s 等
	-true / -false	返回 true / false
	-inum　n	指定 i 节点号，参见“数字”
	-links n	指定链接数，参见“数字”
	-mindepth/-maxdepth d	设置目录搜索的深度
操作	-print	将结果写到标准输出
	-fprint file [format]	将结果[以 format 格式]写到文件 file
	-delete	删除找到的文件
	-exec cmd {} \;	假如 find 的返回值为 True，则将找到的内容替换到{}，执行命令 cmd{}
	-exec cmd {} +	假如 find 的返回值为 True，则将找到的内容添加到{}，最后执行 cmd{}
	-ok cmd {} \;	类似-exec，但执行前提示操作者，回答为 y 或 Y 时则执行 cmd {}
操作符	(expr)	括号运算
	-not/! expr	非运算
	expr1[-a/-and]expr2	与操作
	expr1 -o/-or expr2	或运算

在“-exec cmd {} \;”、“-exec cmd {} +”或“-ok cmd {} \;”中，cmd 和“;”之间至少要有一个空格，{}之内不能有内容（空格也不行）。因为“{}”和“；”是要传递给 find 的，要避免被当前 shell 扩展，需要使用“\”对它们进行转义或用单引号引起来，分别写为\{\}和\;或'{}'和';'。

find 命令的功能的确是用于文件的查找与定位，但它的功能不仅限于此，可以利用它与其他命令配合工作，而完成特定任务。

3．使用示例

（1）按修改时间查找文件。

```
#find /tmp /home -mtime -1 -print        #在/tmp 和/home 下查找 24 小时内没有更改的文件
$ find $HOME -mtime 0                    #在/home 下查找 24 小时内被更改的文件
```

（2）在/home 下查找属于用户 gjshao 的 C 语言程序。

```
#find /home -user gjshao -name "*.c" -print
```

（3）将当前目录内的所有*.c 和*.h 的文件打成 tar 包/tmp/mych.tar。

```
#tar cvf /tmp/mych.tar `find  . -name "*.[ch]" -type f -print`
```

（4）在/home 下查找所有权限为 0644 用户主为 test 的文件。

```
#find /home -perm 0644 -user test -print
```

（5）查找并删除指定文件。

```
#find /tmp -name core -type f -exec rm -f  '{}' \;    #在/tmp 内查找名为 core 的普通文件，并删除之
#find /tmp -name core -type f -ok rm -f  '{}' \;      #查找普通文件 core，并删除之，删除前确认
```

（6）在/sbin 和/usr/sbin 可执行但不可读的文件。

```
#find /sbin /usr/sbin -executable \! -readable -print
```

（7）查找具有特殊属性的文件，并将名字保存到文件中。

```
find  / \( -perm -4000 -fprintf /root/suid.txt "%m %u %p\n" \) , \
    \( -size +100M -fprintf /root/big.txt "%-10s %p\n" \)
##有 suid 属性文件保存到/root/suid.txt，长度>=100MB 者保存到/root/big.txt
```

5.4.2　文件按名查找命令（locate）

1．功能与用法

locate 命令用于按名查找文件，它需要一个或多个数据库支持，这些数据库由每天的例行工作（crontab）程序来建立和更新。因为它是从数据库查找而不是从文件系统实时查找，所以要比 find 命令的搜索速度快，但是由于数据库是非实时更新的，因而也可能会存在以下问题：文件在文件系统中存在但 locate 找不到，或者相反，locate 找到了，但文件却可能已经不存在了。

Locate 的用法为：

```
locate [options] pattern ...
```

2．参数说明

locate 命令的部分参数如表 5-18 所示。

表 5-18　locate 命令的部分参数

参　数	功　能	参　数	功　能
-A | --all	查找所有符合长期保持者	-i | --ignore-case	忽略大小写
-b | --basename	基本名字匹配（与-w 相反）	-l N|--limit=N	限制只输出 N 行
-c | --count	只显示匹配次数	-w|--wholename	完整模式匹配（默认模式，与-b 相反）
-e | --existing	仅查找确实存在者（类似 find）	-r|--regex	使用正则表达式

3．示例

```
#locate passwd            #匹配所有含 passwd 的文件，即匹配*passwd*
#locate -b '\passwd'      #精确匹配 passwd
```

5.4.3　文件复制命令（dd）

1．功能及用法

文件的复制可由 cp 命令来实现，它用于文件或目录的复制是非常有效的，但对设备文件的复制，或创建一个占有连续空间的文件来说最好还是使用 dd。dd 功能是复制指定的输入文件到指定的输出文件，默认时使用标准 I/O。dd 工作时可对文件内容按要求进行转换，还可以从输入文件中有选择地读取，有选择地输出，也可根据不同的设备使用不同大小的 I/O 块，非常高效、方便。dd 命令的用法为：

```
dd [opts=value] …
```

2．参数及说明

dd 命令的常用参数如表 5-19 所示。

表 5-19　dd 命令的常用参数

参　数	功　能	参　数	功　能
if=i_file	指定输入文件，默认为 stdin	of=o_file	指定输出文件，默认为 stdout
ibs=size	指定输入块的大小，单位可是 b，K，M 等	obs=size	指定输出块的大小，单位可是 b，K，M 等
bs=size	同时指定输入/输出块的大小	count=n	只复制 n 块
skip=n	从输入中跳过 n 个块	seek=n	从输出中跳过 n 个块
conv=v	指定转换类型。v=ASCII，block，ebcdic，lcase，ucase，swab 等		

3．dd 示例

（1）构造软盘映像文件。

```
#dd if=/dev/fd0 of=/tmp/fd_img bs=36b #设 I/O 块为 36b，映像文件为/tmp/fd_img
#cp /dev/fd0 /tmp/fd_img              #也可使用 cp 命令，用户可以比较两个命令的执行速度
```

（2）介质复制。

复制软盘或 U 盘等，可以使用临时文件，按以下方法进行：

```
#dd if=/dev/fd0 of=/tmp/mytmpf bs=36k     #构造映像文件，交换磁盘
#dd if=/tmp/mytmpf of=/dev/fd0 bs=36k     #输出映像文件
#rm -f /tmp/mytmpf                        #删除临时文件
```

（3）将第一个 SCSI 硬盘的主引导扇区复制到文件 MBR.sda。

```
#硬盘的设备文件为/dev/sda，主引导扇区位于物理第一扇区
#dd if=/dev/sda of=MBR.sda bs=1b count=1
```

（4）将 file1 转换成大写后写入文件 file2。

```
#dd if=file1 of=file2 conv=ucase
```

5.4.4 链接管理命令（ln）

1．功能及用法

ln 的功能是链接管理，可用于创建文件的硬链接和符号链接。其用法为：

```
ln [options] ... [-T] target linkname          （格式 1）
ln [options] ... target                        （格式 2）
ln [options] ... target ... dir                （格式 3）
ln [options] ... -t dir target...              （格式 4）
```

第一种格式用于创建一个指向 target 的链接 linkname，或在当前目录创建一个指向 target 的同名链接；第二种格式用于在当前目录内容创建一个 target 同名的链接；第三种和第四种格式用于在目录 dir 内创建 target 的同名链接。

2．参数及说明

ln 的部分参数如表 5-20 所示。

表 5-20　ln 的部分参数

参　数	功　　能	参　数	功　　能
-f	强行操作，删除已经存在的同名文件	-i	当链接名已经存在时，交互或提示用户是否删除它
-s	创建符号链接，若不指定此选项则创建硬链接	-b	启用备份。文件重名时，将 name 备份为 name~
-t dir	指定目标目录为 dir	-T target	指定目标文件 target

在创建符号链接时，目标文件可以不存在，但创建硬链接时，因为多个文件可共用一个文件体，使用同一个 i 节点，所以在创建之前被链接的文件必须事先存在。对于目录来说，只能创建符号链接而不能创建硬链接。

3．文件的链接示例

```
#ln -s /tmp/myf my_link          #在当前目录内建立/tmp/myf 的符号链接 my_link
#ln /bin/ls /usr/bin/l           #在目录/usr/bin 内为/bin/ls 创建一个硬链接 l
```

5.4.5 特别文件创建（mknod，mkfifo）

1．功能及用法

用户可以通过文件和目录管理办法来创建普通文件或目录，但是不能用此办法创建设备文件，这是因为设备文件不仅要有一个名字，更重要的是，要通过文件名建立设备与其在内核驱动程序的联系。用于设备文件创建的命令是 mknod，专用于命名管道创建的命令是

mkfifo，它们的用法为：

```
mknod [-m,--mode=MOD] name type [major minor]
mkfifo [-m,--mode=MOD] name
```

name 为要创建的设备文件名；type 为设备类型标志，常用的有 b（块设备）、c（字符设备）和 p（管道设备）；major 和 minor 分别为主、次设备号。

用户还可以在创建设备文件时，通过-m perm 或--mode=perm 为设备文件设置访问权限。当然也可在创建之后，使用 chmod 来修改文件权限。

当用户为某新设备开发了设备驱动程序后，需要用到 mknod 为设备创建一个文件名，以建立设备文件名与设备驱动程序的联系。如果因误操作删掉了某个设备文件，也要参照它原来在系统中的情况使用 mknod 重建它。

设备文件一般放在系统的设备目录/dev 内，但是若用户创建的是自己的私用设备，如命名管道，可以放在自己认为方便的地方。

2．使用示例

（1）创建一个名为 my_FIFO 的命名管道。

```
#mknod my_FIFO p     #创建命名管道 my_FIFO。说明：创建管道时不需要主、次设备号
#mkfifo my_FIFO      #使用专用工具 mkfifo 创建管道
```

（2）在/dev 目录下创建名为 my_device 的字符设备文件，主、次设备号分别为 100 和 0。

```
#mknod /dev/my_device c 100 0
```

5.4.6　磁盘空间和文件系统的使用情况统计（df）

1．功能及用法

df 的功能是按指定格式显示系统中已安装文件系统的使用情况。其用法为：

```
df [ options ] [ filesystem ]
```

若不指定文件系统，则显示系统所有已安装文件系统的使用情况。

2．参数及说明

df 的常用参数如表 5-21 所示。

表 5-21　df 的常用参数

参　　数	功　　能	参　　数	功　　能
-a	显示所有文件系统信息	-i	显示 inode 使用情况
-B sz	指定显示单位	-k	以 K 为单位
-h	按人类易读方式显示	-v	以百分比方式显示相关信息

UNIX/Linux 文件系统的组成部分包括 i 节点区和数据区两大部分，在对文件系统初始化

时，两者的比例是合适的，但对于对文件系统使用的极端情况（若存放的全是大文件或全是小文件）却可能导致 i 节点区和数据区“比例失调”。若全是大文件，则可能首先导致数据区满，若全是小文件却可能首先导致 i 节点区满。使用 df 可以分别监控两者或整个文件系统空间的使用情况，以便对即将满或达到临界值的文件系统管理提出合理的建议。

3．示例

```
#df -v          #显示文件系统的使用情况
#df -i          #显示系统 i 节点的使用情况
```

5.4.7　目录使用磁盘空间情况统计（du）

1．功能及用法

du 的功能是按指定单位统计文件或目录的大小。其用法为：

```
du    [ options ] [ names ]
```

若不指定 names，则使用当前目录。

2．参数及说明

du 的常用参数如表 5-22 所示。

表 5-22　du 的常用参数

参　　数	功　　能	参　　数	功　　能
-a	显示所有文件系统信息	-h	按人类易读方式显示
-b	以字节为单位	-k	以 K 为单位
-B sz	指定显示单位	-s	仅显示指定项目的毛总数，也包括其子目录

3．示例

```
##统计用户 zhangsan 家目录的大小
#du -h /home/zhangsan           #列出每个子目录的大小
#du -hs /home/zhangsan          #列出整个目录的大小
##统计/home 内每个用户目录使用磁盘空间的情况：
#cd /home; du -s `ls`           #ls 用于列目录的内容，其输出作为 du 的输入
```

Linux 还提供有磁盘配额功能用于用户的磁盘空间管理与限制，本书没有介绍，但是仍可以通过 du 来报告每个用户的磁盘使用情况。

5.4.8　数据备份与文件归档管理（tar，cpio）

1．数据备份及任务

系统管理员的一个重要任务就是要确保系统信息的完整性，要做到这一点就必须对系统

中的数据进行经常性备份，万一系统故障或系统中数据出现问题，可以从备份数据中恢复。由此可见，数据备份是保证数据完整的有效办法。

数据备份可分为系统数据备份、用户数据备份和业务数据备份等。

数据备份或系统有多种方式，如完整备份、增量备份和日志备份等。

（1）完整备份就是所需备份的信息全部备份到存储介质，这种办法较安全，因为每次都有完整信息，但是对于日常性备份工作来说，可能会需要大量存储介质。完整备份恢复时将覆盖原有数据。

（2）增量备份是指在某次完整备份的基础上，在以后一个指定时间段内只备份变化部分或新增文件，若某个时间系统数据出问题了，则可以在完整备份的基础上，通过增加备份的数据来恢复数据。增量备份方案不能单独实施，它必须配合完整备份才能保证数据的完整性，并且在使用增量备份的时候，完整备份的时间间隔不能太长。

（3）日志备份也是在完整备份的基础上进行的，它很像增量备份但又不同于增量备份。我们每日的工作都像记账一样，都是在上一笔账的基础上记下一笔，是笔笔相关的流水账。计算机系统的数据变化也是这样，有先后时间和事件上的连续性，所以备份时只需在某个完整备份的基础上记录从那时起以后所发生的事情就可以了。一旦在某个时刻事故发生，则可以在完整备份的基础上根据日志流水逐步重演历史事件，以恢复数据。这也是日志文件系统的设计思路。

数据备份要做到经常化、制度化，备份数据要异地存放。至于采取何种备份方式，则需要根据实际需要或规章制度等来决定。但是为了保证数据的完整性，一个必须坚持的原则是必须保证备份数据的足够性和有效性。所谓足够性，是指在系统或数据出现故障时，所备份的数据一定足够多、足够及时，能完整地恢复所需数据。所谓有效性指的是备份数据的正确性和可用性，不要出现按规定备份了数据，但到恢复数据时却出现备份数据错误或存储介质受损等问题，要经常检查备份数据的有效性。

数据备份可以使用多种不同的存储介质，如硬盘、软件、光盘和磁带等，需根据实际需要进行选择。备份的工具在不同系统中也不尽相同，因系统而异。

在 UNIX/Linux 系统中用于备份的工具有 tar、cpio、dump 和 rsync 等，这里只介绍 tar 和 cpio。

2．磁盘文件归档管理命令（tar）

1）功能及用法

tar 命令的功能是对指定的文件进行归档，从归档文件中取出或恢复指定数据。这里的归档文件或叫档案文件指的是通常意义下的 tar 包（tarball）文件或设备文件。归档或备份过程也就是使用 tar 命令构造 tarball 的过程。被归档的数据可以是文件也可以是目录，在归档或恢复过程中还可以通过 Z、z 或 j 参数进行压缩和解压缩。

归档文件可以是设备文件，如磁带、软盘、硬盘分区等，也可以是普通文件。需要说明的是，当归档文件是设备时，tar 将以裸方式使用它，设备上原来的内容将被覆盖。

不同版本的 UNIX/Linux 系统中 tar 命令的用法和参数可能存在细微差别。tar 的用法可以写为以下形式：

```
tar [ keys ] [ files ]
```

2）参数说明

tar 命令的参数较多，但归纳起来可分为两类，一类是功能参数；另一类是修饰参数。功能参数（如表 5-23 所示）决定 tar 命令的行为，而修饰参数（如表 5-24 所示）则是对功能参数的辅助和修饰。与其他命令不同的是，参数前“-”是可选的。

表 5-23　tar 命令的功能参数

参　　数	功　　能	参　　数	功　　能
-c	创建新文档	-u	只备份归档文件中被修改的文件
-t	显示备份文件的内容列表	-r	在旧档案后追加，不用于普通档案文件
-x	从档案中提取文件，即恢复文件	-d	比较档案内文件与系统内容的差别

表 5-24　tar 命令的部分修饰参数

参　　数	功　　能	参　　数	功　　能
-b N	指定块大小：N*512，默认 N=20	-k	保留旧文件，即不覆盖老文件
-f	指定档案文件名，而不使用默认设备	-T F	从文件 F 中读取要处理的文件名
-h	不归档符号链接，而归档它指向的文件	-v	在工作过程中显示所处理的文件信息
-i	忽略 0 长度文件	-w	强制 tar 以交互式工作
-p	保持权限	-P	采用绝对路径，即保留最前的/
-z	使用 gzip 进行压缩或解压缩	-j	使用 bzip2 进行压缩或解压缩

3）tar 命令示例

（1）使用普通文件作为归档文件。

```
#tar cvf /tmp/mytar [t-z]*                #创建 tar 包文档/tmp/mytar
#tar tvf /tmp/mytar                       #显示 tar 包文档/tmp/mytar 的目录
#tar xf /tmp/mytar                        #从归档文件/tmp/mytar 中提取所有文件
```

（2）使用软盘备份。

```
#tar cvf /dev/fd0 mydata hisdata          #将 mydata hisdata 归档到软盘上
#tar rvf /dev/fd0 /home/gjshao/*.c        #再向软盘上追加/home/gjshao/*.c
#tar xvf /dev/fd0 mydata                  #从软盘上恢复或提出数据 mydata
```

（3）构造和使用压缩 tarball。

```
#tar cvfz /tmp/mytar.tgz  mydir yourdir hisfile    #创建 tgz 格式压缩包
#tar xvfz /tmp/mytar.tgz mydir                     #从压缩文档中提取数据 mydir
#tar xvfz /tmp/mytar.tgz                           #从压缩文档中提取所有文件
```

（4）构造和使用 bzip2 压缩 tar 包。

```
#tar cvfj /tmp/lisi.tbz2 /home/lisi/data           #创建 bzip2 压缩 tar 包
#tar tvfj /tmp/lisi.tbz2                           #查看包中内容
#tar xvfj /tmp/lisi.tbz2                           #从压缩文档中提取所有文件
```

（5）使用磁带机进行数据备份。

```
##使用磁带机/dev/st0 对目录/data 内的数据文件进行完整备份
#tar cvf /tmp/st0 /data
##使用磁带机/dev/st0 对目录/data 中 2 天内修改过的数据文件进行增量备份
#find /data -mtime 2 -print > /tmp/tar_list             #首先得到文件列表
#tar rvf /dev/st0 -T /dev/tar_list                      #增量备份（追加方式）
##增量备份过程可以统一为
#find /data -mtime 2 -print | tar rvf /dev/st0 -T -      #-表示标准输入
```

说明：在备份过程中，当磁盘或磁带满时，备份文件被分隔成多个卷，此时 tar 将提示用户更换新盘或带，在更换时一定要记住介质的先后顺序并做好标记，以便在以后恢复时按顺序使用。

3．文件系统备份与复制命令（cpio）

1）功能及用法

cpio 是 UNIX/Linux 支持的另一类备份工具。和 tar 一样，它的功能也是把指定的文件进行归档，形成归档文件。归档文件可以是设备文件，也可以是普通文件。若归档文件是设备，cpio 也将以裸方式使用它，设备上原来的内容将被覆盖。cpio 所使用的档案格式与 tar 格式不同，两者不能交换使用。

cpio 有三种工作模式，分别是归档模式（copy-out）、提取模式（copy-in）和直接复制模式（copy-pass）。

当以归档模式工作时，它从标准输入读取每行一个的文件名列表，然后根据文件名从系统中读取文件内容，并把它们连同文件内容输出到标准输出上。因此，当使用该模式时常采用 I/O 重定向的办法来进行：将输入重定向为 cpio 提供要备份的文件名列表，将输出重定向将备份结果保存到指定文件中。使用这种模式的典型做法是使用（带有参数-depth 的）find 命令来为 cpio 构造文件名列表，这样做可以绕过目录权限而处理其中的文件，从而可以极小化所遇到的不可读、不可搜索目录而带来的权限问题。

当以提取模式工作时，它将从标准输入上读取归档文件，命令行上所有非参数、非选项内容都将视为归档文件名。它既可提取档案内的文件内容，也可列文件名目录。

在直接复制模式，它按照归档和提取方式将指定文件或目录从一个位置复制到另一个位置，而并不真正使用归档文件。

cpio 命令的基本用法如下：

```
cpio -o [-aBuvV]   [-c] [-O archive ] [ < filelist] [ > archive ]
cpio -i [-ABcdfkqrtuv] [-I archive ] [patterns] [ < archive ]
cpio -p [ -adruv ]   dir < filelist
```

2）参数说明

cpio 的参数也分为功能参数和修饰参数，分别如表 5-25 和表 5-26 所示。

表 5-25　cpio 的功能参数

参　　数	功　　能
-o	归档模式：copy-out

（续表）

参　　数	功　　能
-I	提取模式：copy-in
-p	直接复制模式：copy-pass

表 5-26　cpio 的部分修饰参数

参　　数	功　　能
-a	重置存取时间
-A	向归档文件追加内容，用于归档模式
-d	需要时创建目录
-f	复制与指定模式不匹配的文件，常规情况下是复制匹配文件
-F	指定归档文件名，而不使用标准 I/O
-I archive	提取模式时，指定归档文件名而不使用标准设备
-l	如果有可能，链接文件而不复制它们
--no-absolute-filenames	提取模式时，使用当前路径而非绝对路径
-O archive	归档模式时，指定归档文件名，而不使用标准设备
-q	不显示复制的块数等工作信息
-r	以交互方式，由用户更改文件名
-t	列文档目录（与-i 配合使用）
-u	无条件复制。当覆盖已存在文件时不进行提示

3）cpio 示例

```
#ls *.txt | cpio -ov > /dev/fd0              #将当前目录下的所有*.txt 文件归档到软盘
#cpio -itv < /dev/fd0                        #列出归档文件/dev/fd0 中的内容
#cpio -ir ”*.txt” < myarchv                  #从归档文件 myarchv 中提取文件，并采用交互式
#find . -print | cpio -ov -O /dev/st0        #将当前目录中所有文件归档到磁带/dev/st0
#cpio -itv -I /dev/st0                       #查到磁带/dev/st0 上的归档目录
```

5.4.9　文件的压缩与解压缩

UNIX/Linux 支持多种压缩格式，支持的压缩格式有.Z、.z、.zip、.gz 和.bz2 等，Windows 等操作系统也可支持这些压缩格式。

1．zip 和 unzip

1）功能

以.zip 格式打包压缩或解压缩文件。zip 用于打包压缩，unzip 用于解压缩。

2）用法

```
zip zipfile   file1 file2 …
unzip zipfile   file1 file2 …
```

zip 和 unzip 有很多参数，以上是其基本用法，详细参数请参阅在线手册。

3）应用举例

（1）数据压缩与恢复。

```
#zip myzipfile   file1 file2          #压缩 file1、file2，生成或加入 myzipfile.zip
#unzip myzipfile[.zip]                #解压缩 myzipfile 中的所有文件
#unzip myzipfile[.zip] file2          #解压缩 myzipfile 中的 file2 文件
```

（2）数据打包压缩。

```
#tar cf - . | zip mytar -             #将当前目录内的文件打成压缩 tar 包 mytar
#unzip -p mytar | tar xvf -           #压缩包数据恢复。参数-p 将文件解压到标准输出
#unzip mytar;   tar xvf mytar         #先解压，后提取数据
```

2．gzip/gunzip/zcat

1）功能

以.gz 格式压缩或解压缩文件。gzip 用于压缩文件，gunzip 用于解压缩文件，zcat 用于显示或阅读压缩文件内容。

2）用法

```
gzip   [ -acdfhlLnNrtvV19 ] [ -S suffix ] [ name … ]
gunzip [ -acfhlLnNrtvV ] [ -S suffix ] [ name … ]
zcat [ -fhLV ] [ name … ]
```

默认情况下，gzip 在压缩文件内保存被压缩文件的原名和时间戳，以备解压恢复时使用。gzip 每次只能压缩一个文件，若要打包可与 tar 等命令配合使用。

gunzip 从命令行得到被压缩的文件名，进行解压缩。gunzip 能识别扩展名为.gz、-gz、.z、-z、_z 和.Z 等由 gzip、zip、pack 和 compress 压缩的文件，也能识别.tgz（.tar.gz）和.taz（.tar.Z）等由 tar 包压缩的文件。zcat 等同于 gunzip -c，用于阅读压缩文件。

3）gzip 的部分参数

gzip 的部分参数如表 5-27 所示。

表 5-27　gzip/gunzip 的部分参数

参　数	功　能	参　数	功　能
-a	ASCII 方式	-q	安静方式，屏蔽所有警告输出
-c	输出到 stdout	-r	递归处理目录
-d	解压缩	-S .suf	指定扩展名
-f	强行解压缩，尽管有多个链接	-t	检查压缩文件的完整性
-l	列压缩文件的内容	-#，--fast，--best	调整压缩速度和质量

4）gzip 应用

（1）文件压缩。

```
#gzip myf                      #将 myf 压缩后生成 myf.gz，并删除 myf
#gunzip myf[.gz]               #解压缩 myf.gz，生成 myf，并删除 myf.gz
```

（2）使用 tarball。

```
#tar cvfz mytar.tgz files dirs          #将 files 和 dirs 打包压缩生成 mytar.tgz
#tar tvfz mytar.tgz                     #查看 tar 包 mytar.gz 的内容
#tar xvfz myatr.tgz [ files dirs]       #从 mytar.gz 包提取文件
```

3．bzip2/bunzip2

bzip2 是一个新型压缩工具，比 gzip 等有更高的压缩率。

1）功能

压缩、解压缩文件。

2）用法

```
bzip2 [ -cdfkqstvzVL123456789 ] [ file … ]
bunzip2 [ fkvsVL ] [ file … ]
bzcat [ -s ] [ file … ]
```

3）bzip2 的部分参数

bzip2 的部分参数如表 5-28 所示。

表 5-28　bzip2 的部分参数

参　数	功　能	参　数	功　能
-c	输出到 stdout	-k	保留原文件
-d	解压缩	-t	检查压缩文件的完整性
-f	强行解压缩，覆盖已有文件	-#，--fast，--best	调整压缩速度和质量
-q	默默工作，屏蔽不必要输出		

bzip2 从命令行得到要压缩的文件名，若不指定则从标准输入读文件，并将压缩结果显示在标准输出上。文件压缩后文件名后增加了.bz2，默认情况下要删除原文件，若要保持原文件可使用-k 参数。与 gzip 一样，bzip2 每次只能压缩一个文件，若要打包可与 tar 或 cpio 命令配合使用。

bunzip2 或 bzip2 -d 用于解压由 bzip2 压缩的文件，在解压过程中，解压程序将判断文件的扩展名，并将它转换为相应的类型（见表 5-29）。

表 5-29　bunzip2 解压过程文件类型转换表

输入文件类型	输出文件类型	输入文件类型	输出文件类型
filename.bz2	filename	filename.tbz	filename.tar
filename.bz	filename	filename	filename.out
filename.tbz2	filename.tar		

4）bzip2 的应用举例

（1）文件压缩与解压缩。

```
#bzip2 myfile              #使用 bzip2 压缩文件 myfile，设 myfile 是文本文件
#bunzip2 myfile.bz2        #解压 myfile.bz2
```

```
#bzip2 -d myfile.bz2                     #同上
#bzcat myfile.bz2                        #显示或阅读压缩文件
```

（2）文件的打包与压缩。

bzip2 不能像 gzip 那样可以对文件打包压缩，但可以与 tar 配合使用实现打包压缩。与 tar 配合实现打包压缩的方法有：在 tar 命令行中使用 j 参数；采用管道方式，先用 tar 打包，再由 bzip2 压缩。

① 打包压缩。

```
#tar cvf tarball.tar files dirs                  #打包
#bzip2 tarball.tar                               #压缩。注意扩展名
#tar cvf - files dirs | bzip2 > tarball.tbz2     #打包和压缩过程可统一为此或
#tar cvfj tarball.tbz2 files dirs
```

② 解压与提取 tarball 文件。

```
#bunzip2 tarbll.tbz2                 #先解压 tarball.tbz2
#tar xvf tarball.tar                 #再提取包中所有文件或将解压和展开过程统一为
#bzcat tarball.tbz2 | tar xvf -      #或
#tar xvfj tarball.tbz2
```

5.5 图形界面下的文件和目录管理

很多在字符界面下可以完成的任务，在图形界面下也有相同功能的程序完成类似工作。以下是一些与本章相关的在图形界面下可以实现的功能。

1．文件管理与搜索

用户可通过 Applications→Favorites→Files，或在 CLI 界面直接执行 nautilus 打开“Nautilus”文件管理界面。用户可在此界面的左侧选择工作位置。当位置选定之后，右侧窗体内选中位置的内容。用户可以在此界面控制下完成新建、复制、粘贴、移动、更名、压缩、打开等文件和目录的管理与操作功能。

2．磁盘管理工具

在字符界面下可以通过 fdisk、parted 等对分区进行管理，通过 mount/umount 可以安装或拆卸文件系统，在图形界面下可以通过 Applications→Utilities→Disks 或在 CLI 界面下直接执行 gnome-disks 命令打开磁盘管理工具。

3．磁盘空间管理

在字符界面下有 df、du 等对磁盘空间进行管理，在图形界面下可以通过 Applications→Utilities→Disk Usage Analyzer 可以打开磁盘使用分析器，还可以通过 Applications→Utilities→System Monitor→File System 打开文件系统监视器。

4．备份档案管理

字符界面下的 tar、cpio、zip、gzip 和 bzip2 等打包压缩工具，在窗口界面下可以通过 Applications→Utilities→Archive Manager 或在 CLI 下直接执行 file-roller 打开档案管理器。“Archive Manager”的软件包为 file-roller，在它们的控制下你可以创建和展开几乎所有格式的压缩包，创建新包时的默认格式为*.tar.gz。

5．光盘刻录

在字符界面下通过 mkisofs/genisoimage 构造光盘文件系统，通过 cdrecord/wodim 将光盘映像或数据刻录到光盘，在图形界面下可用的工具有 xfburn 和 k3b 等，用户使用前需要安装它们，方法是：

```
yum install xfburn k3b
```

启动它们的方法是：

“Applications→Accessories→Xfburn”或在 CLI 下直接执行 xfburn 命令；

“Applications→System Tools→K3b”或在 CLI 下直接执行 k3b 命令。

习题 5

1．思考题

（1）umask 的作用是什么？默认情况下新建文件或目录的权限是什么？

（2）何为裸设备，如何使用裸设备？以裸方式使用设备时应注意些什么？

（3）如何将一光盘映像刻录到光盘？

（4）如果某些设备文件被误删除了，如何恢复它们？

（5）数据备份有几种形式，如何做好数据备份工作？

2．单项选择题

（1）设用户所使用的系统硬盘为 IDE 设备且只有一块硬盘，则硬盘上 D:盘所处的分区最有可能是：（　　）

A．/dev/sda5　　B．/dev/hda4　　C．/dev/hda5　　D．/dev/sda4

（2）设用户所使用计算机系统上有两块 IDE 硬盘，Linux 系统位于第一块硬盘上，查询第二块硬盘分区情况的命令是：（　　）

A．fdisk -l /dev/hda1　　B．fdisk -l /dev/hdb2

C．fdisk -l /dev/hdb　　D．fdisk -l /dev/hda

（3）设用户所使用计算机系统只有一个 SCSI 硬盘，则用户在系统开启后首次使用 U 盘时，U 盘所占用的最有可能设备是：（　　）

A．/dev/hda1　　B．/dev/sda2　　C．/dev/hdb　　D．/dev/sdb1

（4）统计磁盘空间或文件系统使用情况的命令是：（　　）

A．df　　B．dd　　C．du　　D．fdisk

（5）构造 iso9660 光盘文件系统的命令是：（　　）

A．dd　　B．mkisofs　　C．cp　　D．cdrecord

（6）用于文件系统直接修改文件权限管理命令为：（　　）

A．chown　　B．chgrp　　C．chmod　　D．umask

（7）有一个文件包 mypkg.tgz，查看其中文件列表的命令是：（　　）

A．tar tvfz mypkg.tgz　　B．cpio -it < mypkg.tgz

C．tar xvfz mypkg.tgz　　D．cpio -ot < mypkg.tgz

3．综合题

（1）综述 Linux 系统的文件权限及其管理。

（2）设 Linux 系统与某 Windows 系统共享硬盘，且 Windows 系统使用的是 FAT 格式文件系统，C:盘位于第二个物理分区上，试以 Windows C:盘为例说明 UNIX/Linux 中文件系统的使用。

（3）tar 和 cpio 命令常用于数据备份，试分别以实例说明如何使用它们进行数据备份和恢复？

（4）试述造成 UNIX/Linux 文件系统受损的常见原因，如何能有效地避免文件系统受损？如何修复受损的文件？

（5）何为映像文件？以一个带有 vfat 文件系统的 3”软盘说明如何创建和使用映像文件？

（6）简述如何对文件系统的空闲空间进行管理。

实验 5

1．权限管理。

2．构造文件系统。

3．文件系统的使用。

4．文件系统的映像创建及映像文件的使用。

5．创建设备文件。

6．数据备份及使用。

7．数据压缩及解压缩。

第 6 章　进程、任务与作业管理

6.1　程序和进程的概念

操作系统的重要任务之一是使用户充分、有效地利用有限的系统资源，也就是说在系统资源一定或有限的情况下，要同时执行更多程序，高效率地完成更多任务。本章的主要内容是从系统的外部来观察系统中与进程（process）和作业（job）或任务（task）相关的行为，并实施某些控制让系统工作得更好，或按照用户的意图来完成指定工作。

6.1.1　程序、进程、作业和任务

程序是一个存储在介质上的，既具有可执行的属性，又具有执行的内容文件。进程是一个程序的动态执行过程。程序的执行要完成某种特定的功能，过程表现为对某个或某些数据的加工和处理。进程突出了一个程序动态的执行过程，而一个程序存储在介质上的文件或数据是一种静态信息。

在多任务、多用户系统中进程具有并发和并行特征。在一个系统内部，同时可以运行很多进程，甚至可以让同一个程序多次同时运行，但它们对应不同的进程，在系统内有不同的进程标识，也就是说不同的进程可以执行同一个程序，但可能使该程序所处理的数据集有所不同。

作业或任务是用户需要计算机完成某项任务时要求计算机所做工作的集合，一个作业可能需要几个程序联合完成。当用户向计算机提交了一个在指定时间执行或重复执行的工作，就可以认为向计算机提交了作业或任务。

6.1.2　调度策略与优先级的计算

在 UNIX/Linux 系统中，每个进程在执行时的优先级是动态变化的，只有优先级最高的进程才能得到运行。当一个进程占有处理机开始执行时，它们的优先级会随着执行时间的增加而逐渐降低，以至于失去处理机。失去处理机的原因可能是多方面的，比如，时间片用完或资源不到位自动放弃处理机，也可能是被优先级高的进程抢占或执行完毕。若它还没有执行完毕，则其优先级将变为“最低”，然后到调度队列的最后排队。随着等待时间的增加，它的优先级逐步增加，在就绪队列中的位置逐步前移，优先级达到最高时才能再次获得处理机而被执行。

在优先级计算过程中有一个静态参数参与计算，这就是传统 UNIX 的 Nice Level 优先级（优先数），按传统做法，把这个参数记为 NICE。系统为每个进程设置了一个默认的 NICE 值（0），通过命令 nice 或 renice 命令可以调整进程的 NICE 值，从而达到调整优先级的目的。在 Linux 系统中，它的调整范围为−20～19（其他 UNIX 中的 NICE 取值范围可能与此不同），沿用 UNIX 的传统，NICE 的值越大，进程的优先级越低，得到执行的机会越少。

6.1.3　Linux 操作系统的启动

Linux 系统的启动是在其引导程序（引导块或逻辑引导扇区）被装入后在该引导程序的

控制下进行的。首先初始化系统内部数据结构，如构造空闲缓冲区、区表结构、页表项等，然后将根文件系统安装到根“/”下，并创建系统的 0#进程、设置它的运行环境，至此系统的内核已经启动完成，但此时系统还不能做太多工作，至少还缺少与用户的交互部分。为此，系统继续引导，创建 1#进程，然后由 1#进程做进一步初始化工作，直到启动完毕，各种服务开始运行，终端用户可以登录。

1#进程继续初始化过程是按照初始化配置文件进行的，这个配置文件在早期时为 /etc/inittab，现在在不同系统中已经有了各自的变化。启动配置文件的内容用于规定启动哪些服务或管理进程，为设定终端生成登录子进程，然后等待用户在终端上登录，整个启动过程如图 6-1 所示。

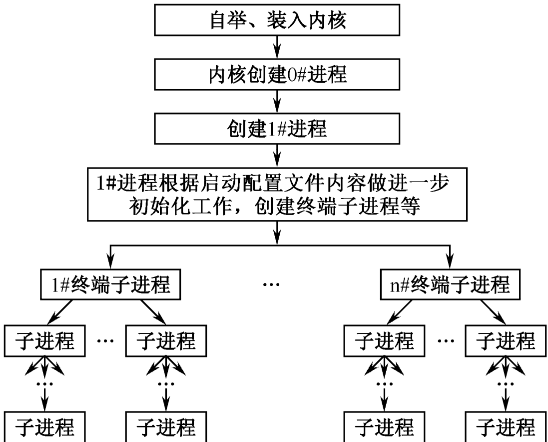


图 6-1　Linux 系统的启动过程

6.1.4　0#进程与 1#进程

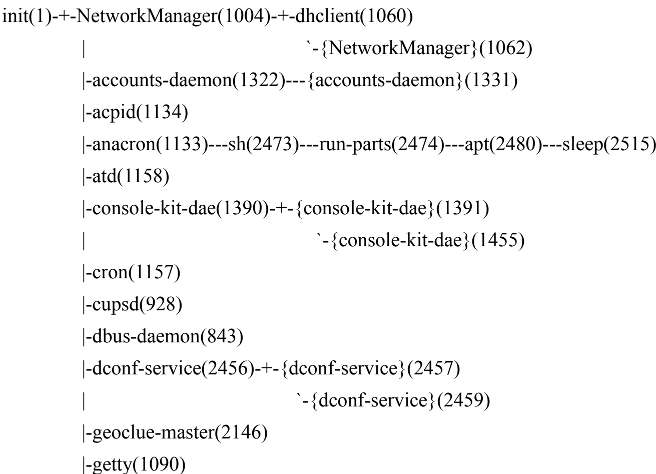
0#进程与 1#进程是 UNIX/Linux 系统中两个最重要的进程。

在 UNIX 系统中 0#进程是唯一只在核心态下执行的进程。它的功能有三点：调度分配处理机；负责进程交换；初始化时创建 1#进程。在 Linux 系统中，0#进程在创建出 1#进程后变成了空闲进程（Idler），当系统中没有其他进程就绪时它才运行，而它原来的功能被分散在几个内核进程中。

1#（init 或 systemd）进程是系统启动时创建的创建进程的进程。它的主要作用是根据启动配置文件的内容初始化系统、创建系统运行所需的进程。在 Linux 中，还要根据运行级别触发运行级别初始化事件，执行相应的脚本程序进行初始化。系统初始化完成后，1#进程变成了回收进程，专门领养没有父进程的孤儿进程或回收状态为 ZOMBIE 的僵尸进程。

从进程创建关系来看，0#进程创建了 1#进程，它是 1#进程的父进程。1#进程在系统启动过程中创建了系统所需要的其他进程，这些被创建的进程是 1#进程的子进程，而子进程又可创建属于自己的子进程，因此系统中除了 0#进程外，1#进程是其他所有进程的祖先进程。

Linux 系统的进程之间的家族关系可用命令 pstree 来查看，图 6-2 为某 Linux 系统的进程树的局部（来自 pstree -p 的输出）。



```
init(1)-+-NetworkManager(1004)-+-dhclient(1060)
        |                      `-{NetworkManager}(1062)
        |-accounts-daemon(1322)---{accounts-daemon}(1331)
        |-acpid(1134)
        |-anacron(1133)---sh(2473)---run-parts(2474)---apt(2480)---sleep(2515)
        |-atd(1158)
        |-console-kit-dae(1390)-+-{console-kit-dae}(1391)
        |                       `-{console-kit-dae}(1455)
        |-cron(1157)
        |-cupsd(928)
        |-dbus-daemon(843)
        |-dconf-service(2456)-+-{dconf-service}(2457)
        |                     `-{dconf-service}(2459)
        |-geoclue-master(2146)
        |-getty(1090)
```

图 6-2　某 Linux 系统的进程树

6.1.5　进程状态及查询（ps）

进程在执行过程中不是一气呵成的，而是走走停停在不同的状态间转换，比如，进程的三个基本状态为执行、就绪和阻塞。ps 命令可用于检查系统内的进程状态。

1．功能与用法

ps 的功能是显示进程的状态和信息，给出系统当前正在运行进程信息的快照，可以配合 kill 结束系统中失控或不必要的进程。ps 支持多种 UNIX 系统格式的个性化显示方式，其用法为：

```
ps [options]
```

2．参数描述

ps 命令的参数繁多，部分参数如表 6-1 所示。

表 6-1　ps 命令的部分参数

参　数	功 能 说 明	参　数	功 能 说 明
-A，-e	显示系统内的所有进程	-a	显示所有与终端相关的进程（但不包括会话组长）
a	显示当前终端的所有进程	-c	显示 CLS 和 PRI 栏
c	列出进程名，但不包含路径名	-C pname	显示进程名为 pname 的同名进程信息
-d	显示所有进程（但不包括会话组长）	e	在命令名后显示环境变量
-f	显示带有路径的进程名和进程参数	f	用 ASCII 字符显示树状结构的进程间关系
g，-g/-G gid	显示与进程组 gid 相关的信息	h	不显示标题
-H	显示树状结构以表示进程间的关系	-j, j	以作业控制方式显示进程信息
-l, l	以长格式显示进程信息	L	显示标题栏信息
-N	显示除执行 ps 终端以外的所有进程	p/-p pids	显示指定进程的信息
T/r	只显示本终端所有/正在执行程序	t/-t tty	显示 tty 相关进程信息，t 用于显示当前终端进程信息
u, U/-U uid	显示与用户 uid 相关的信息	v	以虚拟内存方式显示进程状态
x	显示所有进程信息，不区分终端	Z/-Z/-M	显示进程与 SELinux 相关的信息

3．使用示例

```
#ps -el                  #长格式显示所有进程的信息
#ps -Af | more           #带命令及参数显示系统中的所有进程信息，并分屏显示
#ps -t /dev/tty1         #查询与终端 tty1 相关的进程信息。可简略为 ps -t 1 或 ps -t tty1
#ps -f -u gjshao         #查询与用户（gjshao）相关的信息
#ps -axj                 #查询整个系统内的进程信息情况
```

4．进程的状态

在 ps 输出信息中的状态标志如表 6-2 所示。

表 6-2　ps 输出信息中的部分进程状态信息

状　态	说　　明
R	Running/Runable，进程在执行中或在就绪队列排队
D	Delaying，非中断性睡眠，通常是因为等待 I/O
S	Sleep，正在睡眠，等待事件发生
T	Traced or stopped，被跟踪或停止
Z	Zombie/Defunction，僵尸状态。进程已经终止，但父进程没有对它的结束做适当处理
W	Wait，无内存页面（被换出）（kernel 2.6.xx 后不再使用）
N	低优先级进程
L	Locked，页面被锁进内存（用于实时控制或用户控制的 I/O）
<	高优先级进程

6.1.6　三类进程

UNIX/Linux 系统是多用户、多任务操作系统，在系统内同时运行着很多进程，如系统管理进程、服务器进程和用户进程等，这些进程大致可以分为三类。

1．前台进程

前台进程（foreground process）是指用户直接控制的用于完成某个任务的进程，因此也叫终端交互式进程。它从标准输入读数据，向标准输出写数据，将错误信息输出到标准错误，也可以是用户直接交互控制的完成某种功能的程序，如文字加工与处理、游戏、浏览器和不同行业的各种应用程序等。

2．后台进程

后台进程（background process）是指在系统后台运行的进程。在一个系统中可能运行着很多一般用户不知道、也不关心的进程，如系统本身用于管理和控制的进程、各种服务器进程等。前台进程也可放在后台运行，但这时要用到 I/O 重定向。

守候进程（daemon process）也叫服务器或精灵进程，它是后台进程的一种，该类进程永

久不停地运行着，以等待其他进程提出服务请求而为它们提供服务。

3．批处理进程

批处理进程（batch process）是用户按照某种意图将一批作业和任务通过编程的方法提交给系统，让系统在某个合适的时间来调度和执行的进程。批处理进程是通过 shell 编程等方法，将所要处理的工作规划好，之后由系统启动并依次执行的进程。事实上，是在某个 shell 的控制下解释执行的。

6.2　登录 shell 的启动与定制

当 Linux 系统启动完毕后，在活动终端上为用户启动登录进程，并等待用户登录。在用户登录时要输入用户名和密码，当用户名和密码通过系统认证后，系统启动 shell 为用户提供人机交互功能，默认情况下系统出现提示符为#（超级用户）或$（一般用户）。一旦出现了 shell 提示符，用户就可以输入命令及命令所需要的参数，由 shell 解释并执行之。当用户完成操作任务后，可以通过注销的办法退出 shell，系统再回到登录界面。

6.2.1　用户登录过程与登录 shell 的启动

1．登录过程与启动控制文件

登录 shell 的启动流程和工作过程如图 6-3 所示。

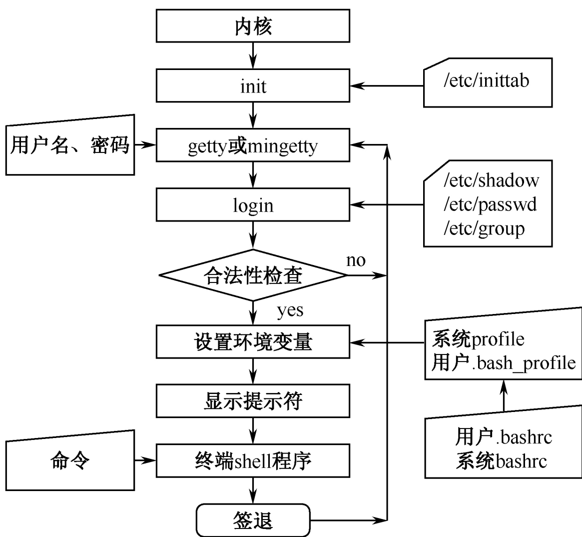


图 6-3　shell 的启动过程

用户登录过程中，在输入用户名和密码后，系统要检查以下文件。

（1）/etc/passwd：系统用户数据库。

（2）/etc/shadow：影子密码和用户登录控制文件。

（3）/etc/group：系统组定义文件。

如果用户名和密码合法，则启动交互 shell，否则要求用户重新输入用户名和密码。

2．登录 shell 的启动

在系统启动 shell 过程中，依次进行以下操作。

（1）执行系统 profile 文件/etc/profile（并在它的控制下执行/etc/profile.d/目录下的*.sh 脚本文件）。

（2）执行用户 profile 文件~/.bash_profile（Ubuntu 用的是~/.profile）。

（3）如果~/.bashrc 存在则执行之。

（4）如果/etc/bashrc 文件存在则执行之。

需要说明的是，以上步骤都是在当前 shell 内执行的（参见 shell 编程部分），目的是为用户设置环境变量或做相关准备。一切执行完毕之后出现系统提示符。

6.2.2 用户登录控制与 shell 的定制

根据用户登录和 shell 的启动过程可以看出，用户可在 shell 的启动控制文件（/etc/profile，~/.bash_profile 或~/.profile，/etc/bashrc 和~/.bashrc）中加入自己的内容，而影响用户 shell 的启动过程，如设置或添加自己所需的环境变量等。如果要在这些文件中加入用户自己的内容，一般放在文件的最后，这样可以使用户在登录之后就进入某个特定程序，当其退出时用户也随之注销。因此，可让用户只在某个规定封闭环境下工作，而不让其真正进入系统，看不到系统的提示符，从而可以提高系统的安全性和操作的方便性。当然，这需要一定的 shell 编程知识。

登录 shell 定制，除了以上对 shell 启动过程的控制之外，还可以直接修改用户的 shell 程序，这可以在用户创建时完成，也可以在用户创建后处理。

1．修改用户登录控制文件

用户的登录控制文件为~/.bash_profile 或~/.profile、~/.bashrc，可以修改其中之一来影响登录 shell 的启动。一般修改的是~/.bash_profile（红帽）或~/.profile（Ubuntu），至于怎样修改，添加或去除什么内容，要根据实际需要。例如，可加进一个特定程序~/.bash_profile 或~/.profile 文件的最后，使得在用户登录时，在一切都准备之后执行之（参见 shell 编程部分）。

2．修改用户的登录 shell

shell 有多种，用户可以从中选择适合自己需要的。大家可能已经注意到了，useradd 和 usermod 命令都有一个-s 选项，这个选项是用来选择 shell 的。也就是说，可以使用-s SHELL 选项，在创建用户或修改用户时为用户指定一个 shell 而非使用默认的 bash 或 bsh。例如：

```
useradd -s /bin/csh test          #创建 test 用户，并指定登录 shell 为 csh
usermod -s /bin/csh test          #修改 test 用户，使其登录 shell 为 csh
```

另外 Linux 还提供了一个命令 chsh，意思是改变用户 shell，可以用它来为用户修改登录 shell，方法是：

```
chsh -s SHELL USER                #将用户 USER 的登录 shell 修改为 SHELL
```

USER 为用户名；SHELL 为带有路径的 shell 程序名。在执行过程中，若 SHELL 不在/etc/shells 中存在，则 chsh 会发出警告，但并不影响修改。使用示例如下：

```
chsh -s /bin/tcsh test                    #将用户 test 的登录 shell 修改为/bin/tcsh
```

6.3 Linux 系统的启动过程分析

Linux 系统的启动软件包已经经历了 SysVinit、upstart 和 systemd 三个阶段。SysVinit 来自早期的 SysV UNIX 系统，是启动管理基础。upstart 可以理解为是 SysVinit 的改进与发展，现在还在应用于 Ubuntu 系统，但 Ubuntu 系统 15.04 后已经引入 systemd，现在是两者并存。在红帽家庭中，自从 Fedora 15 开始引入 systemd 后，最新版本的 Fedora、RHEL 和 CentOS 中都使用了 systemd 进行启动管理，并允许与 SysVinit 并存。

6.3.1 SysVinit

1．/etc/inittab 文件

1）/etc/inittab 文件结构

/etc/inittab 是系统初始化配置文件之一，它描述在系统启动时哪个进程将被启动。在/etc/inittab 文件中以#开始的行为注释行，空行不起作用，其他的为有效行。有效行结构为：

```
id：runlevels：action：process
```

（1）id：1～4 个字符长的唯一标识（在有的系统内被限制在 1～2 个字符），而对于终端线路来讲，其值为对应的线路号。

（2）runlevels：运行级别（参见 init 命令）。

（3）action：在 runlevels 运行级别应该采取的动作。

（4）process：采取 action 时所执行的程序。

部分有效的 action 域值及意义如表 6-3 所示。

表 6-3　常见的 action 域值及意义

值	意　　义
respawn	进程将重新启动，当 process 死掉时系统将立即重新启动它
wait	当系统进入某级别时进程将运行一次，且在运行期间，init 将等待它结束
once	当系统进入某级别时进程将运行一次
boot	在系统启动时执行该命令，此时的运行级别将被忽略
bootwait	在系统启动时执行，init 将等待该进程的结束，其中的运行级别将被忽略
off	什么都不做，对于终端线路来讲，相当于没有被激活
ondemond	当某个进程设置该标志时，启动指定运行级运行时将被执行
sysinit	系统启动时执行。它将先于 boot 或 bootwait 执行，其中的运行级别将被忽略
initdefault	设置系统启动的默认级别
powerwait	当 UPS 关闭时执行，init 将等待此进程执行完毕

（续表）

值	意　　义
powerfail	当 UPS 掉电时执行，但 init 将不等待此进程执行完毕
powerokwait	当 UPS 恢复时执行
powerfailnow	当 UPS 电池将空时执行此命令
ctrlaltdel	当系统收到信号 SIGINT，即用户同时按下 ctrl_alt_del 时 init 执行该行规定的程序，通常是重新启动

2）/etc/inittab 示例

某 Linux 系统/etc/inittab 内容如下：

```
#设置默认运行级
id:5:initdefault:
#系统初始化
si::sysinit:/etc/rc.d/rc.sysinit
#处理不同运行级的脚本
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
… … … … …
l6:6:wait:/etc/rc.d/rc 6
#处理 ctrl+alt+del
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
#处理电源故障
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
#处理电源恢复
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
#为多用户级（2345）启动终端进程
1:2345:respawn:/sbin/mingetty tty1
… … … … …
6:2345:respawn:/sbin/mingetty tty6
#为第 5 级启动 X-Windows
x:5:respawn:/etc/X11/prefdm -nodaemon
```

2．与启动过程相关的文件和目录

除了/etc/inittab 外，与系统初始化相关的还有其他文件和目录。

1）启动脚本文件汇总目录 init.d

所谓启动脚本文件汇总目录，是指存放系统启动所需脚本文件的目录，在红帽系统下为：

/etc/init.d 或/etc/rc.d/init.d　（前者是后者的一个符号链接）

在 ubuntu 下仅有：

/etc/init.d

该目录的每一个文件负责一个功能或服务的启动与关闭管理，且文件名与服务名密切相关，比如，halt 用于系统的关闭、atd 和 crond 分别用于 at 和 crontab 的作业调度、vsftpd 用于 vsftp 服务器的启动与停止等。

2）不同运行级的启动和停止脚本

Linux 系统可以运行在不同的级别，每个运行级都有自己的启动和停止脚本目录。该目录在红帽系统为：

/etc/rcN.d，/etc/rc.d/rcN.d （0≦N≦6，为不同的运行级）

在 ubuntu 下仅为：

/etc/rcN.d （0≦N≦6，为不同的运行级）

不同目录的内容对应该运行级初始化所使用的脚本。其中的每一个脚本文件都是 init.d 内某个脚本文件的符号链接，用于对某个服务或管理程序的启动或关闭。

rcN.d（0≦N≦6）目录内文件排列是按照该级别初始化顺序排列的。系统启动时，对应级别启动目录内的启动脚本是按脚本文件名排列顺序由低到高依次执行的，因此 rcN.d（0≦N≦6）内文件命名的排列顺序是相当重要的，否则可能破坏系统内进程间的相互依赖关系。因此文件名必须按顺序排列正确。

每个 rcN.d 目录都是由 K 和 S 开头的脚本文件组成的，K 开头的文件用于停止，S 开头的文件用于启动，其他类型的脚本文件将被忽略。

为使目录 rcN.d（0≦N≦6）内的文件名按启动顺序排列正确且具有实际意义，文件名一般采用[KS]MNinitdname 的形式，文件的开头字符为 K 或 S 用于标明停止或启动，MN 为两位整数用于排列顺序，initdname 为对应 init.d 内的脚本文件名。例如，在 rc5.d 有脚本程序：

```
lrwxrwxrwx    1 root root     15 Oct 29 15:19 K35dhcpd -> ../init.d/dhcpd
lrwxrwxrwx    1 root root     18 Oct 29 15:19 S99rc.local -> ../init.d/rc.local*
```

前者用于 dhcpd 服务的停止，后者用于 rc.local 的启动。

在同一个 rcN.d（0≦N≦6）目录内可能既有启动脚本又有停止脚本，停止脚本用于对切换前一运行级进程的处理。由于 K 排列在 S 之前，所以在对某一个运行级初始化时总是先完成停止旧运行级的工作或任务，然后再进行新运行级的初始化。

用户启动某个任务的脚本可使用以上办法进行处理，也可以使用直接将脚本放在相应运行级的办法来实现。按启动顺序，脚本程序命名时应将其放在系统启动脚本的后面。用户启动的服务也可在停止目录内进行相应停止处理。更常用的方法是将用户启动脚本追加在/etc/rc.d/rc.local 或/etc/rc.local 文件的尾部。

3）不同运行级脚本执行控制程序

不同运行级脚本执行控制程序为：

/etc/rc，/etc/rc.d/rc（前者是后者的符号链接），在 Ubuntu 下为/etc/init.d/rc

由/etc/inittab 文件的内容可知，系统在进行运行级切换时，总是通过此脚本控制不同运行级脚本的执行，方法是：

lN:N:wait:/etc/rc.d/rc N （0≦N≦6）

例如：

```
l5:5:wait:/etc/rc.d/rc 5
```

4）红帽系统/etc/rc.d

在红帽系列系统中，将存放在/etc 下的 init.d、rc.local、rc0.d~rc6.d 等集中存放在/etc/rc.d 之内，然后再通过在/etc 下创建同名符号链接的办法实现与其他系统的兼容。命令

```
#ls -l -d /etc/rc*
```

输出将告诉用户结果。

5）用户进程启动脚本

本地用户进程启动脚本为：

```
/etc/rc.local，/etc/rc.d/rc.local
```

用于本地用户定义程序的启动。若用户需要在系统启动时启动某个自己的程序，则可以将启动命令放在该脚本里。对此文件的修改将会在各真正的（0、1 和 6 外）运行级有效，因为在目录 rc2.d～rc5.d 中均有一个文件 S99local 指向该文件。

这里需要提醒的是，加入的部分程序应该是瞬间可以结束的，或让加入的部分在后台运行，否则系统的初化或启动过程将因加入程序的运行而无法继续。

6.3.2　upstart

从形式上看，upstart 是在 sysVinit 基础上发展起来的，因为使用了 sysVinit 几乎所有的目录和配置文件，但是 upstart 拆分了/etc/inittab 文件，将其中每一行的功能变成一个事件脚本文件，然后集中存放在一个叫事件目录的地方（该目录在红帽系列系统中为/etc/event.d，在 ubuntu 下为/etc/init，为了描述方便，我们把这个目录称为事件目录，统一记为 EVENT_D）。于是系统初始化时，由原来的扫描/etc/inittab 文件变成了扫描事件目录。

在 upstart 负责启动管理的系统中，/etc/inittab 的唯一作用是设置默认运行级。其中的关键一行内容是：

```
id:DEFAULT_LEVEL:initdefault:          #DEFAULT_LEVEL 为默认运行级，可取 1~5。比如
id:5:initdefault:                      #DEFAULT_LEVEL=5。默认启动级别为 5
```

若/etc/inittab 不存在，或 DEFAULT_LEVEL 未定义，则桌面或工作站系统的默认运行级为 5。

1．事件目录的内容

事件目录中的每个文件对应一个初始化事件。事件脚本处理文件与原 inittab 文件内容的对应关系如表 6-4 所示。

表 6-4　事件目录的内容及与原/etc/inittab 的关系

事件处理脚本文件	功　　能	与原/etc/inittab 的对应关系
rcS	系统初始化事件处理	si::sysinit:/etc/rc.d/rc.sysinit
rc0	运行级别 0 初始化事件处理	l0:0:wait:/etc/rc.d/rc 0
…	…	…
rc6	运行级别 6 初始化事件处理	l6:6:wait:/etc/rc.d/rc 6

（续表）

事件处理脚本文件	功　　能	与原/etc/inittab 的对应关系
tty1	终端 F1 事件处理	1:2345:respawn:/sbin/mingetty tty1
…	…	…
tty6	终端 F6 事件处理	6:2345:respawn:/sbin/mingetty tty6
serial	串口控制台事件处理	
prefdm	X 终端启动事件处理	x:5:respawn:/etc/X11/prefdm -nodaemon
control-alt-delete	Ctrl_Alt_Del 事件处理	ca::ctrlaltdel:/sbin/shutdown -t3 -r now

说明：

（1）脚本文件的命名方式在红帽系统与 ubuntu 之间有差异但不大，表现在所有文件名在 ubuntu 都多了个后缀.conf，如红帽的 rcS 对应 ubuntu 的 rcS.conf。

（2）在 ubuntu 中，rcS.conf 用于 single-user 模式初始化。

（3）红帽的 rcS 用于执行传统的 sysv-rc 启动脚本，而在 ubuntu 中，rc-sysinit.conf 用于完成此项工作。

（4）在红帽中，0～6 级别的初始化脚本分别为 rc0~rc6，但在 ubuntu 中，统一为 rc.conf。

2．事件脚本文件结构

（1）节（stanzaz）。

在 events 脚本文件中包含以下节：

console {logged|output|owner|none}：定义任务输出的处理方式。logged-定向到日志文件；output-定位到标准输出；owner-进程变为 console 的所有者，可以从键盘接收信号；none-定位到/dev/null。

chroot <path>：设置 chroot 路径，比如：chroot /var/named。

chdir <path>：设置目录。比如：chdir /var/named。

exec　<command> <arguments>：执行命令，比如：exec /bin/true。

env <name>=<value>：定义环境变量，比如：env LD_LIBRARY_PATH=/opt/app/lib。

daemon：设置 daemon 标志（包含 service），任务将在后台运行。

nice <value>：设置 nice 值，比如：nice 5。

pre-start、pre-stop、post-start 和 post-stop：处理事件间的依赖关系。事先或事后启动与停止。

respawn：若终止，则重新启动。

respawn limit [count [timeout]]：设置重启限制。若在 timeout 秒内重启次数超过了 count 次，则将自动停止不再重启。limit 的默认值为 5 秒重启 10 次。

script　...　end script：定义 shell 脚本块，参见 shell 编程部分。

start on <event>：事件 event 到来时执行，例如

start on startup：系统启动后立即执行。

tart on stopped rc3：事件'rc3'停止后开始。

stop on <event>：事件到来时停止。例如

stop on runlevel 0：进入第 0 级时停止；

stop on stopping dbus ok：'dbus'停止后停止。

umask <umask>：定义 umask，比如：umask 0755。

（2）事件（EVENTS）。

系统的两个核心事件分别是 startup 和 shutdown。除此之外，还有由 startup 和 shutdown 事件引起的其他事件，也有用户触发的事件。常见事件如下：

control-alt-delete：信号 ctrl+alt+del。

kbdrequest：键盘事件发生。

power-status-changed：接收信号 SIGPWR，电源状态发生了变化。

runlevel <LEVEL>：初始化到指定级别 LEVEL。

starting <job>：作业 job 正在启动。

started <job>：作业 job 已经成功启动。

stopping <job>：作业 job 正在停止。

stopped <job>：作业 job 已经成功终止。

stalled：作业没有执行也没有未决信号。

3．基于事件机制的系统启动过程控制

在事件机制控制下的系统启动过程如下：

（1）加电启动内核；

（2）内核启动 init；

（3）确定运行级别 N；

（4）触发 runlevel 事件，开始运行/etc/event.d/rcN；

（5）/etc/event.d/rcN 执行/etc/init.d/rc N；

（6）执行/etc/rcN.d/内的脚本文件；

（7）最后启动成功，给出登录界面。

4．事件目录的使用和系统启动控制处理示例

忽略 ctrl+alt+del 信号。

用于对 ctrl+alt+del 信号控制的文件为 control-alt-delete[.conf]，其内容为：

```
#control-alt-delete - emergency keypress handling
##This task is run whenever the Control-Alt-Delete key combination is
#pressed.   Usually used to shut down the machine.
start on control-alt-delete
task
exec /sbin/shutdown -r now "Control-Alt-Delete pressed"
```

由此可以看出，信号 ctrl+alt+del 到来时的默认行为为执行命令

```
/sbin/shutdown -r now "Control-Alt-Delete pressed"
```

若要改变这个行为，只需要修改此控制文件，将其中的 exec 所执行的命令改为其他命令即可。若要忽略 ctrl+alt+del 信号，只需将命令改为任何一个不起实际作用的命令，如/bin/true 或/bin/false 等就可以。现在，将事件处理文件的最后一行修改为：

exec /bin/true

然后，运行命令“init q”或重启系统后就生效了。

由于 upstart 正在被 systemd 所取代，不再给出更多这方面的示例。

6.3.3 systemd

systemd 是本书所用系统的启动管理和服务管理工具，可用于本书所涉及的所有系统中。在引入 systemd 的系统中，init 命令被替换成 systemd，telinit 命令被替换成 systemctl。当系统启动时，systemd 作为 1#进程运行，它的作用是初始化系统，并管理系统和用户空间的各种服务。为了与 Sysvinit 兼容，如果 systemd 作为 init 调用且 PID≠1，则它执行 telinit（systemctl），且不加修改地传递所有的命令行参数给 telinit。

出于对 SysVinit 的兼容，对用户来讲，sysvinit 的内容及系统启动脚本和各种目录的命名方式等仍然可以沿用，对于以前的 SysV 的传统服务仍然可以采用传统方法进行管理。

1．systemd 的 Unit

systemd 提供一个被称为“unit（单元）”实体的依赖关系系统。系统初始化需要做的事情非常多，systemd 将启动过程中的每一步都抽象为一个配置单元。比如，一个服务是一个配置单元，一个挂载点是一个配置单元，一个交换分区的配置是一个配置单元等，但也有一些 Unit 是系统通过其他配置或系统状态自动创建的。

一个 unit 可以是关于 service、socket、device、mount point、automount point、swap file、partition、start-up target、file system path 或 timer 等的。

1）常用 unit

（1）service unit：服务 unit，用于服务控制，形如*.service，如 crond.service、xinetd.service。

（2）socket unit：IPC 或网络通信 unit，IPC 或网络通信控制，形如*.socket，如 cups.socket、rpcbind.socket。

（3）target unit：组合 units，用于提供系统在启动过程中的同步点，形如*.target，如 default.target、poweroff.target。

（4）device unit：内核设备发现 unit，用于实施基于设备的激活，形如*.device，如 dev-cdrom.device、dev-cdrw.device、dev-sda.device。

（5）mount unit：文件系统中的安装点 unit，形如*.mount，如 boot.mount、proc-fs-nfsd.mount。

（6）automount unit：用于为需要安装者提供自动安装能力，形如*.automount，如 proc-sys-fs-binfmt_misc.automount。

（7）timer unit：定时触发器 unit，用于基于定时器的定时触发，启动其他活动（Activities），形如*.timer，如 systemd-tmpfiles-clean.timer、systemd-readahead-done.timer。

（8）path unit：路径 unit，用于当文件系统对象变化或被修改时启动其他基于路径的活动（activattion），形如*.path，如 systemd-ask-password-wall.path。

2）unit 的常见状态

在 unit 被用于启动或关闭的处理过程中，可以处于“active（活动）”（意为启动、绑定、插入等，取决于 unit 的类型），或“inactive（不活动）”（意为停止、非绑定、被拔出等）状

态，也可处于两者之间的正在进行状态“activating（正在激活）”和“deactivating（正在禁止）”。一个类似于 inactive 特殊的状态“失败（failed）”也是允许的，这种状态在服务失败（进程结束时返回错误码，崩溃或超时）时进入。如果进入了这种状态，则原因将被记入日志文件。需要说明的是，各种 unit 状态还可以划分为一些子状态，它们可以被映射到以上 5 种常见状态。

3）unit 间的关系

在 systemd 处理某个引导或启动事务时，诸多的 units 间存在着正需求（Requires=）、负需求（Conflects=）和顺序依赖（After=和 Before=）等关系，systemd 知道并能处理这些依赖关系。

（1）正需求关系（Requires=units）：若本 unit 活跃，则它所需求的 units 也必须活跃，如果 Units 中的一个变得不活跃（deactivated）或失败了，则本 unit 也将变为不活跃状态（deactivated）。

（2）负需求（Conflects=units）：本 unit 和 units 不能共存。若本 unit 启动，则 units 必须停止，反之亦然。

（3）顺序依赖关系（After=units）要求两者都要启动，且本 unit 先于 units 启动；同样地，顺序依赖关系（Before=units）要求两者都要启动，且本 unit 后于 units 启动。

除以上的关系之外，还有 Wants=、RequiresOverridable=、Requisite=、RequisiteOverridable=、BindTo=、OnFailure=、PropagateReloadTo=、PropagateReloadFrom=、OnFailureIsolate=、IgnoreOnIsolate=和 StopWhenUnneeded=等。

4）常见 systemd unit 的功能及作用

常见 systemd unit 的功能及作用如表 6-5 所示。

表 6-5　常见 systemd unit 的功能及作用

unit	说　明
basic.target	一个包含早期启动的 target unit。systemd 将自动地将它添加到 sysV 的 runlevel1.target～runlevel5.target 中的 Requires 和 After 依赖关系中。通常，作为基本的初始化需要，也会添加到其他 daemon 服务的 unit 配置文件的 Requires 和 After 中
ctrl-alt-del.target	用于处理 Control+Alt+Del 信号，通常作为符号链接到 reboot.target
dbus.service	用于处理 D-Bus 的 unit
default.target	systemd 的系统启动默认 unit。通常通过符号链接指向某一个启动 target，如 multi-user.target 或 graphical.target
display-manager.service	显示管理服务。通常作为符号链接指向 prefdm.service 或其他类似的显示管理服务
emergency.target	紧急启动策略 unit，对应在控制台（console）上启动应急 shell。此多用于启动参数的修改，并以 systemd.unit=emergency.target 方式添加到内核启动参数中
graphical.target	图形界面启动策略，用于启动进入第 5 级 X-windows
halt.target	系统关闭策略 unit。当一个应用程序需要关闭系统时启动此策略
multi-user.target	多用户启动策略（非图形界面，对应 runlevel3）
poweroff.target	系统关闭策略（对应 runlevel0.target）
reboot.target	系统重启策略（对应 runlevel6.target）
rescue.target	单用户或系统维护模式启动（runlevel1.target）
runlevel0.target	关闭系统策略，链接到 poweroff.target。对应第 0 级
runlevel1.target	系统维护模式策略，链接到 emergency.target。对应第 1 级

（续表）

unit	说　明
runlevel2.target~runlevel5.target	对应运行级 2～5 启动策略
shutdown.target	服务关闭策略，用于系统关闭时关闭服务。systemd 将以 Conflicts 依赖关系添加此策略到停止系统的 unit 中（如运行级 0 或 6）
sigpwr.target	UPS 电源控制策略，当电源失效时启动此策略
sysinit.target	systemd 特殊 unit，它覆盖了早期的启动脚本。systemd 自动添加 Wants=和 After=依赖关系到 SysV 的 0～6 级运行策略
swap.target	交换设备管理策略，用于管理交换区

5）配置目录

systemd 用到的主要目录如下。

本地配置目录：/etc/systemd/system。

运行 unit 目录：/run/systemd/system。

已经安装软件包的 unit 位置：/lib/systemd/system。尽管红帽的 systemd 手册页说是在/usr/lib/systemd/system，但由于/lib 是/usr/lib 的一个符号链接，故在红帽系统中两者是相同的。

2．管理工具（systemctl）

1）功能及用法

systemctl 用于控制 systemd 的系统和服务管理。这里只简单介绍一些选项和主要功能，用于控制系统的启动和随后介绍的服务管理。systemctl 的用法为：

```
systemctl [options] {COMMAND} [NAME …]
```

其中，options 为选项；COMMAND 为子命令；NAME 为被操控的对象。

2）选项

systemctl 有很多选项，部分常用选项如表 6-6 所示。

表 6-6　systemctl 的部分常用选项

选　项	功　能
--help，-h	帮助
--type=，-t	指定类型，如 service、target 和 socket 等
--all，-a	列所有 unit
--failed	只列失败的 unit
--quiet，-q	抑制标准输出
--no-pager	不分屏/页显示
--no-reload	当与 enable/disable 配合使用时，不重新加载配置文件
--force，-f	当与 enable 配合使用时，覆盖已有链接；当与 halt、poweroff、reboot 和 kexec 配合使用时，不关闭所有 unit 而直接关闭系统
--runtime	当与 enable/disable/is-enabled 或其他命令配合使用时，只临时修改配置，而不保存到磁盘，对下次启动无效
--full=	完整显示 unit 的信息
--root=	使用此指定 unit 文件搜索路径

（续表）

选　　项	功　　能
--kill-who=	与 kill 配合使用时，指定被杀进程。默认为所有
--signal=，-s	与 kill 配合使用时，指定信号

3）子功能

systemctl 可实现很多子功能，部分子功能如表 6-7 所示。

表 6-7　systemctl 的子功能

命　　令	功　　能
list-units	列出所有已知的 unit
list-unit-files	列出所有已安装的 unit 文件
list-jobs	列出正在运行的 JOB
is-active [NAME …]	检查指定 unit 是否在活动。若最后一个是活动的返回 0，否则非 0
status [NAME …|PID …]	检查指定 unit 或进程的运行情况
show [NAME …|JOB …]	显示 unit 或 JOB 的属性信息
start/stop/restart [NAME …]	启动/停止/重启服务
reload [NAME …]	重载指定（而非 systemd 的）服务配置文件
reload-or-restart [NAME …]	重载或重启指定服务
try-restart [NAME …]	试图重启指定服务，若服务没有运行则不起作用
reload-or-try-restart [NAME …]	重载或试图重启
isolate [NAME …]	启动指定及相关的 unit，其他 unit 将被终止。仅当 AllowIsolate=true 时才能生效
kill [NAME …]	向 unit 中的一个或多个进程发送信号。用--kill-who=指定进程，--kill-mode=指定 mode，--signal=指定信号
reset-failed [NAME …]	重置指定处于 Failed 状态的 unit，若不指定 NAME，则重置所有处于 Failed 状态的 unit
is-enabled	检查指定 unit 是否已经启用。若是则返回 0，否则显示启用情况
enable/disable/reenable/preset [NAME …]	启用/禁用/重新启用指定服务或 unit
mask [NAME …]	将指定 unit 链接到/dev/null。使得它不能再启动
unmask [NAME …]	做与 mask 相反的工作
cancel [NAME …]	撤销指定作业
daemon-reload	重载 systemd 管理器配置。这将重载所有 unit 文件，重新创建依赖关系树。不要将此与 load 和 reload 相混淆
daemon-reexec	重新运行 systemd 管理器。调试时使用
show-environment/ set-environment [NAME=VALUE …]/ unset-environment [NAME …]	显示/设置/取消 systemd 的环境变量
default	进入默认模式，等价于运行 default.target
rescue	进入 rescue mode，等价于运行 isolate rescue.target
emergency	进入 emergency mode，等价于运行 isolate emergency.target
halt	关机/关闭系统，等价于运行 start halt.target 或命令 halt

（续表）

命　　令	功　　能
poweroff	关闭系统且关闭电源，等价于运行 start poweroff.target 或 poweroff 命令
reboot	重启系统，等价于运行 start reboot.target 或 reboot 命令
get-default	获得当前的默认启动 unit（可比较 runlevel 命令）
set-default UNIT	设置默认的启动 unit 为 UNIT

说明：enable 只启用而不启动；start 只立即启动而不启用；disable 只禁用而不停止；stop 只停止而不禁用。reenable 等价于先禁用再启用；restart 相当于先停止再启动；preset 则将 unit 初始回默认状态。

3．应用示例

1）列所有 unit

```
#systemctl list-units                    #列出所有 unit，并分屏显示
#systemctl --no-pager list-units         #列出所有 unit，不分屏显示
```

说明：systemctl list-units 功能只能显示 systemd-unit，而不能显示 SysV 服务，而 chkcongfig 却只能显示 Sys V 服务，而不能显示 systemd-unit，不过两者配合却可以得到全部。

```
#systemctl systemctl list-unit-files     #列出已经安装的所有 unit 文件
```

2）关闭系统

```
#systemctl poweroff                      #使用 poweroff.target 关闭系统，等价于 poweroff 或 init 0
```

3）重启系统

```
#systemctl reboot                        #使用 reboot.target 重启系统，等价于 reboot 或 init 6
```

4）切换到系统维护模式

```
#systemctl rescue                        #切换到系统维护模式，等价于 init 1
```

6.4 服务管理

6.4.1 systemctl

系统有很多服务，这里仅以对 vsftpd 服务管理来说明使用 systemctl 管理服务的过程。要管理一个服务，必须要先知道服务的名字。vsftpd 服务器的服务名为 vsftpd.service。

vsftp 服务器软件包为 vsftpd，如果系统中还没有安装，则首先安装它，方法是：

```
#yum -y install vsftpd                   #红帽系统
#apt-get -y install vsftpd               #ubuntu 系统
```

1）列出或检查 vsftpd.service 的状态

```
#systemctl | grep vsftpd                  #列出所有 units 从中过滤 vsftpd 服务
#systemctl is-enabled vsftpd.service      #检查是否已激活
```

2）激活 vsftpd 服务

```
#systemctl enable vsftpd.service          #若成功返回 0，否则非 0
```

3）检查 vsftpd 运行状态（注：此时仍未启动）

```
#systemctl status vsftpd.service          #检查状态
```

可根据输出判断服务的当前状态。若在输出中有“loaded”、“inactive”和“dead”等内容，说明已被激活，但可能还没运行，也可能会加载失败。

4）立即启动/停止/重启/重载服务

```
#systemctl start/stop/restart/reload vsftpd.service          #成功时返回 0，否则为非 0
```

5）禁用 vsftpd 服务

```
#systemctl disable vsftpd.service                            #成功时返回 0，否则为非 0
```

6.4.2 早期的服务管理

1．红帽的早期服务管理

用户还可以直接使用命令 chkconfig、service 或服务启动脚本在终端上或程序中直接对服务进行简单、快捷的管理。

1）chkconfig

chkconfig 用来检查和设置系统各种服务的开机自启动，其用法为：

```
chkconfig [--list] [name]
chkconfig [--level levels] name <on|off|reset>
chkconfig [--level level] name
chkconfig --add | --del | --override name
```

参数 name 为服务名；选项--list 用于服务列表；--level levels 用于指定运行级别；--add、--del 或--override 用于添加、删除或覆盖一个已有服务（本书不讨论后三个功能）。

（1）显示系统服务列表，以及这些服务在每个运行级别是否启动。

```
#chkconfig --list                                            #显示所有服务启动状态
```

输出结果（的局部）如图 6-4 所示，前面是独立服务，尾部是基于 xinetd 的服务。

（2）显示单个服务的启动状态。

如果用户欲检查某单个服务（如 xinetd）的信息可使用以下两种形式：

```
#chkconfig --list xinetd                  #显示 xinetd 的情况
#chkconfig --list | grep xinetd           #同上
```

```
livesys          0:off  1:off  2:off  3:on   4:on   5:on   6:off
livesys-late     0:off  1:off  2:off  3:on   4:on   5:on   6:off
netconsole       0:off  1:off  2:off  3:off  4:off  5:off  6:off
network          0:off  1:off  2:off  3:off  4:off  5:off  6:off
xinetd based services:
    chargen-dgram:     off
    … … … ..
    time-stream:       off
```

图 6-4　chkconfig 命令的部分输出

（3）设置或取消某服务（如 nfs）在指定运行级别的自启动。

```
chkconfig --level 345 nfs off          #设置在运行级 345 中开机不启动 nfs 服务
chkconfig --level 3 nfs on             #设置在运行级 3 中开机自启动 nfs 服务
```

（4）检查指定服务在指定运行级别的开启状态。

```
chkconfig   --level 3 nfs              #检查 nfs 服务在第 3 级是否为开机自启动
chkconfig   nfs                        #检查 nfs 服务在当前级是否为开机自启动
```

此种方式只能指定一个级别，若不指定则使用当前级别。检查后并不输出任何信息，而是以返回码告诉使用者：可用 echo $?查询返回码，若返回码为 0 表示为 on，否则为 off。

2）service

service 命令用来改变服务的当前状态，其用法为：

```
service SCRIPT cmd [options]
service --status-all
service --help | -h | --version
```

参数 SCRIPT 为服务名或/etc/init.d/内与服务同名的脚本文件名；cmd 一般可取的值有 start、stop、restart、status 和 reload 等分别用来启动、停止、重启、服务状态检查和修改配置后重新载入，具体有多少或是什么取决于脚本文件的内容和功能。例如，要启动、停止或重新启动 httpd 服务，则可以分别使用命令：

```
#service httpd start
#service httpd stop
#service httpd restart
```

3）启动脚本

系统的所有服务启动脚本文件均存放在/etc/init.d/或/etc/rc.d/init.d/目录，与服务名相同。比如，httpd 服务的启动脚本文件名为 httpd，网络服务的启动脚本名为 network。若要启动、关闭或重启一个服务，只采用以下形式就可以了。

```
#/etc/init.d/SCRIPT cmd [options]              #或
#/etc/rd.d/init.d/SCRIPT cmd [options]         #此行仅用于红帽系列
```

比如，按此方法重启网络服务的命令为：

```
#/etc/init.d/network restart
```

此种方法与 service 相同，只是在使用 service 命令时，不需要指定脚本所在位置而已。

4）图形界面

红帽系列系统还提供有服务图形界面管理工具，它们分别是 system-config-services 和 ntsysv，使用前需要先安装它们。安装方法是：

```
#yum install system-config-services ntsysv
```

用户可以在 CLI 界面下通过命令

```
#system-config-services
#ntsysv [--level <LEVELS>]
```

启动它们，然后可根据需要进行操作。

2．ubuntu 的早期管理

1）service

ubuntu 的 service 用法与红帽的 service 相同。

2）sysv-rc-conf

sysv-rc-conf 是 ubuntu 的 sysv 服务配置和管理工具，也是对 ntsysv 的改进，但是默认不被安装，若要使用它，则须事先安装之，方法如下：

```
#apt-get install sysv-rc-conf
```

sysv-rc-conf 的用法如下：

```
sysv-rc-conf    --list    [service]
sysv-rc-conf [ --level levels ] service <on|off>
```

有点类似红帽的 chkconfig，更多示例不再介绍。

3）图形界面

当不带任何参数运行 sysv-rc-conf 时，可以进入简单的“图形”界面，用户可在此界面下对服务的启动进行管理。

4）启动脚本

可以按照红帽系统办法使用启动脚本进行启动管理，但要注意脚本文件的命名在不同系统间有差异。比如，网络服务管理脚本在红帽系统下为 network，在 ubuntu 下为 networking。

6.4.3　超级服务器（xinetd）

从原理上讲，服务器进程的工作就是打开并监听一个端口，并且等待客户连接，如果客户端来访，响应客户请求并为客户提供服务。但是，现在操作系统中，一般都有很多服务，若为每个服务都创建一个进程进行监听，就会浪费系统资源，尤其对于那些不常使用、很少使用，但有可能被使用的小服务，如 chargen、daytime、discard、echo 和 time 等更是如此。于是，引入了“超级服务器”或“委托服务器”的概念。

“超级服务器”可以为多种服务管理连接，当收到连接请求时，它能够确定连接请求受理所需的程序并创建相应的进程来为客户提供服务。

1．软件安装与服务管理

Linux 系统的“超级服务器”为 xinetd.service，归属 xinetd 软件包，使用前需先安装它。安装方法是：

```
#yum install xinetd                                    #红帽系统
#apt install xinetd                                    #Ubuntu 系统
```

可通过以下方法管理超级服务器 xinetd.service:

```
#systemctl enable/disable xinetd.service               #启用/禁用 xinetd.service 服务
#systemctl start/stop/restart/reload xinetd            #启动/停止/重启/重载 xinetd 服务
#systemctl status xinetd.service                       #启用/禁用 xinetd.service 服务
```

2．配置文件（xinetd.conf 与 xinetd.d/*）

xinetd 的配置文件为/etc/xinetd.conf 和/etc/xinetd.d/*，前者是主配置文件，后者为一个个由主配置文件包含的被管理服务的子配置文件。主配置文件是配置 xinetd 服务本身或各子服务的通用部分，而各被管理服务的子配置只是用于配置子服务器自身的。

默认情况下，/etc/xinetd.conf 为其所管理的服务提供默认配置，其中，“#”开头的是注释行，有效内容大致为：

```
defaults
{
    log_type      = SYSLOG daemon info
    log_on_failure     = HOST
    log_on_success     = PID HOST DURATION EXIT
    cps           = 50 10
    instances     = 50
    per_source    = 10
    v6only             = no
    groups             = yes
    umask         = 002
}
includedir /etc/xinetd.d
```

其中，常用的配置项及意义如下：

（1）defaults{…}：为所有 xinetd 服务指定默认值。

（2）instances：指定一个 xinetd 服务的最大实例数。

（3）log_type：设置 log 的类型。

（4）log_on_success：定义连接成功时，log 记录的内容。

（5）log_on_failure：定义连接失败时，log 记录的内容。

（6）cps：定义服务率上限，由两个数组成，第一个为每秒处理的服务个数，如果超过此值，则服务会暂时被 disabled；第二个数定义下次被重新激活须经过的秒数。

（7）includedir /etc/xinetd.d：定义子服务配置文件所在目录为/etc/xinetd.d/。

/etc/xinetd.d/内的每个文件定义了一个子服务，文件名一般与服务名相同，文件内容的格

式也相同，大致如下：

```
service <服务名>
{
<属性> <操作符> <值>
… … …
}
```

其中 service 是必需的关键字。子服务的常见属性如表 6-8 所示。

表 6-8　xinetd 配置文件中的常用配置项

配　置　项	意　　义
disabled	取值范围为 yes 或 no：yes-禁用；no-启用
flags	定义服务标识，可取值范围为 INTERCEPT、NORETRY、NODELY、IPv4、IPv6 等
socket_type	定义 socket 类型，可取值范围为 stream、datagram、raw 等
protocol	定义协议类型。可取值范围为必须在/etc/protocols 中定义，常用的有 TCP、UDP 等
wait	并发或重复：yes-阻塞（重复）型材；no-并发型
user/group	定义服务进程的用户主/组
Server/server_args	带有完整路径的服务器守护程序名/命令行参数
type	定义服务类型，可取值范围为 RPC、INTERNAL、TCPMUX/TCPMUXPLUS 和 UNLISTED
only_from/no_access	只能从/不能从那些主机或 IP 接收请求

当一个 xinetd 服务及相应软件包被安装之后，会在/etc/xinetd.d/内安装其服务器子配置文件，但也有特殊情况，比如，不提供子配置文件，或虽有子配置文件，但不放在/etc/xinetd.d 内。

若一个服务在/etc/xinetd.d/安装有子配置文件，则直接可以使用；若一个服务提供有可供被托管运行的子配置文件，但又不在/etc/xinetd.d/内，则只需要配置文件复制到/etc/xinetd.d/，并让文件名与服务名相同就可以了。比如，vsftpd 提供的子配置文件为/usr/share/doc/vsftpd/examples/INTERNET_SITE/vsftpd.xinetd，可将此文件复制到/etc/xinetd.d/并更名为 vsftpd。若没有提供子配置文件，则需要参照其他服务配置文件和服务属性手工制作一个，或从其他系统中借用一个，并稍作修改。

下面为 telnet 服务配置文件/etc/xinetd.d/telnet 的内容。

```
service telnet
{
    flags          = REUSE                #表示当中断或重启 xinetd 时，TCP/IP Socket 可重用
    socket_type = stream                  #表示使用 TCP 的 Socket 类型
    wait           = no                   #表示该服务提供并发功能
    user           = root                 #设置进程的 UID（由 root 用户操作）
    server              = /usr/sbin/in.telnetd   #设置服务程序
    log_on_failure      += USERID         #表示当连接失败时，系统还要记录用户 ID
    disable             = yes             #表示不允许 xinetd 启动本项服务。yes-不启用；no-启用
}
```

所有被托管服务的子配置文件内都有一行：

```
disable       = yes/no                    #no-启用；yes-不启用
```

用于表示该服务是否被启用。如果要启用该服务，则需将其中的“yes”改为“no”，然后重启或重载 xinetd 服务就可以。

对于 telnet 服务，只需将上述/etc/xinetd.d/telnet 代码中的 yes 改 no，然后启动或重启 xinetd 服务就可以。

6.4.4 rc-local.service

Linux 系统提供有 rc-local.service，用于本地用户自定义服务启动。rc-local.service 所在位置为/lib/systemd/system/rc-local.service，其内容在红帽和 Ubuntu 系统中有所不同，可通过命令

```
# cat /lib/systemd/system/rc-local.service
```

查看其内容，其中的关键部分是

```
ExecStart=/etc/rc.d/rc.local start          # 红帽系统
ExecStart=/etc/rc.local start               # ubuntu 系统
```

由此可以看出，rc-local.service 要用到 rc.local 文件。

rc.local 文件在红帽系统为/etc/rc.d/rc.local，且事先不存在，在 ubuntu 系统为/etc/rc.local，已经存在。用户若让系统启动时自动启动自己定义服务，只需要将启动脚本添加到 rc.local 文件的最后即可。在早期的系统中确实是这样，但在引入 systemd 之后，除此之外，还要启用 rc-local.service，方法是

```
# systemctl enable rc-local.service         # 红帽&Ubuntu
```

比如，若要让系统启动时删除/tmp/*.tmp 文件，则可以编辑 rc.local 文件，在最后添加如下一行内容：

```
rm -rf /tmp/*.tmp & 2>/dev/null &
```

说明：命令的后台执行很重要，也就是说命令行最后的“&”很关键。假如启动的不是一个立即能够结束或永远不会结束的程序，将会导致后面的程序无法继续启动，从而导致系统无法正常启动。

6.5 进程管理与调度命令

6.5.1 可执行文件的 setuid、setgid 权限和目录的 sticky 属性

在 UNIX/Linux 系统中，用户级别和执行权力是相关的。但是有些工作，如修改/etc/passwd 和/etc/shadow 文件等，对于普通用户也是必需的，因为它要修改自己的密码，但是系统中这些文件对于普通用户是不允许有写操作的，于是出现了让普通用户通过某种机制行使超级用户权限的问题。

1．suid 和 sgid 权限

1）概念

对于可执行文件，有些文件还有另外两个与执行有关的属性。

（1）设置用户 ID 权限：setuid/suid。

（2）设置组 ID 权限：setgid/sgid。

当一个程序具有 suid 属性时，它执行时的 uid（有效 uid）将是该程序所有者的 uid，记为 euid，而执行者原来的 uid 称为真实 uid，记为 ruid。此时，程序的执行者具有程序所有者的权限。

当一个程序具有 sgid 属性时，它执行时的 gid（有效 gid）将是该程序的 gid，记为 egid，而执行者原来的 gid 称为真实 gid，记为 rgid。此时，程序的执行者具有程序所有者所在组的权限。

suid/sgid 属性只对二进制可执行文件有效，而对可执行的脚本文件无效。

当一个属于超级用户的可执行二进制程序具有 suid 属性时，在一般用户执行它时，也就相当于有了超级用户的身份和权限。同样，当一个属于超级用户所在组的可执行二进制程序具有 sgid 属性时，当一般用户执行它时，也就相当于有了该组员的身份和权限。由此可见，suid 和 sgid 权限的存在可能会给某些程序的执行带来方便，因为权限被放大了，但这也是一种危险，因为过大的权力可能会给系统的安全带来隐患，因此设置这种权限的可执行二进制程序不宜太多。

常用的设置 suid 的程序有/bin/ping、/usr/bin/passwd、/usr/bin/newgrp、/usr/sbin/suexec、/usr/bin/sudo 和/usr/sbin/traceroute 等被公用，但又必须具有较高权限的程序。

2）管理

按照文件的权限，若用八进制表示，则它的 u、g、o 的读、写和执行权限可分别表示为 400、40、4，200、20、2 和 100、10、1，对应于 suid 和 sgid 的权限则为 4000 和 2000。

suid 和 sgid 对应的权限也是用 chmod 来管理的。设 myp1、myp2、myp3 的权限分别为 -rwxr-xr-x(755)、-rwxrwxr-x(775)和-rwxrwxrwx(777)，则可用以下方法设置 suid 和 sgid 的权限。

（1）字符方式。

```
#chmod u+s myp1          #为 myp1 设置 suid。修改后的权限为 rwsr-xr-x
#chmod g+s myp2          #为 myp2 设置 sgid。修改后的权限为 rwxrwsr-x
#chmod ug+s myp3         #为 myp3 同时设置 suid 和 sgid。修改后的权限为 rwsrwsrwx
```

（2）数字方法。

```
#chmod 4755 myp1         #为 myp1 设置 suid
#chmod 2775 myp2         #为 myp2 设置 sgid
#chmod 6777 myp3         #为 myp3 同时设置 suid 和 sgid
```

2．sticky 位

1）概念

在 4 位八进制权限的最高 1 位所表示的 3 位二进制中，已经被 suid 和 sgid 占去高 2 位，还剩下最后 1 位，它可以用来描述目录或文件的另一个属性。当用于描述目录时，称为限制

删除位（restricted deletion flag），用于描述可执行文件时，称为粘着位（sticky bit）。为了描述方便，我们把它统一称为 sticky 位或 sticky 权限。

当一个目录具有 sticky 权限时，它内部的文件只能被它的所有者或超级用户删除、更名或移动。当一个可执行文件具有 sticky 权限时，在早期的系统中，程序执行时的正文段映像将被锁在交换区中，可使得程序装入更快速。

设置粘着位的目录有/tmp、/usr/tmp 和/var/spool/uucppublic 等。

2）管理

sticky 位的属性值为 1000，可通过 chmod 命令设置。设有目录 mydir 和文件 myfile 的权限为 755，则可通过以下方法来设置它的 sticky 位。

```
#chmod +t mydir myfile          #字符方式。修改后的权限为 1755 或 rwxr-xr-t
#chmod 1755 mydir myfile        #数字方式
```

3．查找

用户可以使用 find 命令查找系统具有 suid、sgid 和 sticky 权限的文件，方法如下：

```
find / -perm /1000 -print       #仅查找具有 sticky 的位者
find / -perm /2000 -print       #仅查找具有 sgid 的权限者
find / -perm /4000 -print       #仅查找具有 suid 的权限者
find / -perm /6000 -print       #仅查找具有 suid 和 sgid 的权限者
find / -perm /7000 -print       #查找具有 suid、sgid 和 sticky 的权限者
```

4．说明

当对一个在执行位没有权限的文件添加 suid、sgid 和 sticky 权限时，相关对应的字符表示不是 s 和 t，而是 S 和 T。比如，root 用户有一个名为 x 的文件，其他权限为 644（rw-r--r--），在执行

```
#c hmod +s,+t x
```

命令后，其权限为 7644，符号表示为 rwSr-Sr-T。用户可使用 ls -l 查看 x 的权限。

6.5.2　进程管理与调度命令

1．按名称向进程发信号或终止进程的执行（killall）

1）功能及用法

kill 用于向指定进程或进程组发送指定信号（见表 6-9），如果未指定信号，则默认发送 TERM（15）。若接收者收到信号，而没有对信号捕获并进行特殊处理，则它将被终止运行，也就是被杀掉了。

killall 是另一个用来向指定进程发信号或终止进程的命令，它不像 kill 使用进程的 PID 作为操作对象，而是使用进程名。由于在系统中可能有多个同名进程同时运行，因此使用 killall 终止进程时可能会有扩大化的问题。其用法为：

```
killall [-e] [-g] [-i] [-q] [-v] [-w] [-V] [-s sig] [--] name …
killall -l
```

killall 发送的信号可以是信号名或值。

2）参数说明

killall 命令的部分参数如表 6-9 所示。

表 6-9　killall 命令的部分参数

参　数	意　　义	参　数	意　　义
name	进程名。信号接收者	-l	列出可用信号
-i	交互式工作，终止进程前需要用户确认	-I	忽略大小写
-g	终止指定所属组内的所有进程	-u user	指定用户
-q	安静模式，当不存在指定进程时也不报错	-s sig	指定信号（参见-l 或表 3-38）
-e	精确匹配进程名（默认 15 个字符后不再区分）	-v	报告信号是否发送成功
-w	等待所有被杀进程结束。若被杀进程忽略发送信号或处于僵尸状态，则会造成 killall 无限等待		

3）使用示例

```
#killall -l            #查询可用信号
#killall man           #终止所有名为 man 的进程
#killall -9 bash       #终止所有名为 bash 的进程，将造成所有终端用户退出系统
```

2．根据进程名等属性查找进程和发送信号（pgrep/pkill）

1）功能及用法

pgrep 的功能是在系统内部查找满足指定指标的进程，并按格式要求输出搜索到进程的 pid 等信息。pkill 也是要搜索进程，但不是为了显示，而是向它们发送信号。其用法为：

```
pgrep [options] pattern
pkill [options] pattern
```

pgrep 和 pkill 在搜索时要求搜索指标全部满足，比如：

```
#pgrep -u root sshd
```

搜索到的将是用户 root 的 sshd，而非其用户的。

```
#pgrep -u root,daemon
```

搜索到的是用户 root 和 daemon 的所有进程。

2）参数及说明

pgrep 与 pkill 的常用参数如表 6-10 所示。

表 6-10　pgrep 和 pkill 的常用参数

参　　数	意　　义	参　　数	意　　义
pattern	指定命令及命令行参数（扩展正则表达式）	-signal	指定信号（默认信号为 SIGTERM）（仅 pkill 使用）
-a|--list-full	列 pid 与命令行（仅 pgrep 使用）	-l|--list-name	列 pid 与进程名（仅 pgrep 使用）

（续表）

参　　数	意　　义	参　　数	意　　义
-c	仅显示匹配的进程数量（仅 pgrep 使用）	-d delim	指定输出进程 pid 的分隔符（默认为换行）
-n	仅搜索（刚启动的）最新进程	-o	与-n 相反，仅搜索启动最早的旧进程
-x	完全匹配进程名（-f 时完全匹配命令行）	-f	搜索整个命令行（默认仅搜索命令名）
-g pgrp,…	仅在进程组 pgrp,…中搜索	-G rgid,…	仅搜索真实 gid 在 rgid,…中的进程
-P ppid,…	仅搜索父进程在 ppid,…中的进程	-s sid,…	仅搜索会话 id 在 sid,…中的进程
-u euid,…	仅搜索有效 uid 在 euid,…中的进程	-U ruid,…	仅搜索真实 uid 在 ruid,…中的进程
-t term,…	仅搜索控制终端在 term,…中的进程	-v	反向匹配

3）示例

```
#pgrep vsftd              #搜索 vsftpd 进程，并显示其 pid
#pgrep -a vsftd           #搜索 vsftpd 进程，并显示其 pid 及命令行参数
#pgrep -u root named      #搜索 root 用户的 named 进程
#pgrep -d, -x httpd       #显示所有 httpd 进程的信息，输出分隔符为逗号
#pkill -HUP syslogd       #向 syslogd 发送信号 HUP，让其重新加载配置文件
#pkill -9 bash            #终止所有名为 bash 的进程（将造成所有使用 bash 的终端用户退出）
```

3．确定使用指定文件或文件系统的进程（fuser）

1）功能和用法

在系统管理和 shell 编程时经常需要确定使用某个设备或文件系统的进程和用户的情况，fuser 可以胜任此工作，其用法为：

```
fuser [-fuv] [-a|-s] [-4|-6] [-c|-m|-n space] [ -k [-i] [-M] [-w] [-signal] ] name ...
fuser -l
fuser -V
```

2）参数说明

fuser 命令的部分参数如表 6-11 所示。

表 6-11　fuser 命令的部分参数

参　　数	功 能 说 明
-k	终止所有使用设备或文件系统的进程
-i	在杀死进程前提示确认，若无-k 则忽略此参数
-l	列出所使用的信号名
-m	name 指定的是一个安装的文件系统
-n namespace	在指定名字空间内搜索，名字空间可是文件、UDP 和 TCP 等
-s	默默工作，不显示输出信息，但可以使用其返回值
-signal	指定信号，而非默认的 SIGKILL
-u	显示用户信息
-4/-6	只搜索 IP4/IP6 的 sockets 信息。默认情况下将同时搜索两者

在 fuser 的默认输出中，进程号的后面可以跟有 c、e、f、r、m 等字符，意义为：c—当前目录；e—进程正在执行；f—打开文件（默认情况下不输出此标志）；r—根目录；m—内存映像文件或共享库。

3）应用示例

（1）杀死所有使用某设备的进程。

有时需要让使用某设备或文件系统的所有进程终止，例如，要把使用终端/dev/tty5 的用户踢出系统，就可采用终止使用该设备所有进程的办法来实现。

```
#fuser -k /dev/tty5
```

（2）显示使用本地网络端口的进程。

从网络监控的角度来讲，有时需要了解系统中哪个端口正在被哪些进程或用户使用，有多少个进程在使用。

```
#fuser telnet/tcp ftp/tcp          #监视本地 telnet 和 ftp 端口的 tcp 活动情况
#fuser -u 23/tcp 21/tcp            #监视本地 telnet 和 ftp 端口的 tcp 活动情况，并添加用户
#fuser -n tcp 21                   #监视本地 ftp 端口的 tcp 活动情况
```

4．让进程抗 SIGHUP 运行（nohup）

在 UNIX/Linux 环境下进程可在前台运行，也可在后台运行。但不论哪种情况，只要启动进程的用户退出了或终端线路断路了，那么他创建的所有进程都将被终止，这是因为这些进程都将收到 HUP 信号。按照系统规定，收到 HUP 信号的进程都将被终止。为了保证用户退出后所启动的后台进程仍能继续运行，就要求进程对信号 HUP 具有免疫能力。使用 nohup 启动的进程对信号 HUP 就具有免疫力。nohup 命令的用法为：

```
nohup command [ arguments ]
```

后台进程是不能和终端交互的。如果输入没有被重定向，则定向到/dev/null，如果输出没有被重定向，则它将输出追加到当前目录的 nohup.out 文件。如果该文件没有写权限，则输出将被重定向到$HOME/nohup.out。如果两者均无写权限，则命令不能被执行。如果有写权限且 nohup.out 不存在，则将创建。在创建时不对同组和其他人赋予访问权（权限为 600）。若文件已经存在，则不改变原文件的存取权限。程序未重定向的标准错误也将被追加到 nohup.out 文件。

nohup 不能直接创建后台进程，创建后台进程应在回车前添加“&”符号。尽管没有输出，但我们仍可以通过其返回值来确定其执行情况。返回值及意义为：125—nohup 自己失败；126—命令找到但不能执行；127—命令找不到；其他—所执行命令的返回值。

5．设置进程的优先级（nice）

1）功能与用法

nice 的功能是设置进程的（静态优先数）NICE 值，并启动进程，从而改变进程获得调度的机会。对于那些不急迫的任务，可以通过调高它的 NICE 值来降低其优先级，以保证紧急任务的执行，当然也可以指定一个负的 NICE 值来提高进程的优先级。nice 命令的用法为：

```
nice [options] [cmd [arg … ]]
```

当不带参数或变量运行 nice 时，将显示默认的 NICE 值，如果用 nice 启动某个进程，而没有指定 NICE，则将 NICE 设置为 10，从而降低了优先级。如果要设置负的 NICE，从而提高进程的优先级，则要求用户必须具有超级用户权限。关于 NICE 设定可以使用以下形式：

（1）-n<NICE>：　　例如:-n5，NICE 值为 5；-n-5，NICE 值为-5

（2）-<NICE>:　　例如:-5，NICE 值为 5；--5，NICE 值为-5

（3）--adjustment=<NICE>　　例如:--adjustment=5，NICE 值为 5

2）使用示例

```
#设有程序 myp（如 cat）可以在前、后台运行
#myp&                  #以默认优先级运行
#nice myp&             #以默认优先数（10）降低 myp 的运行级
#nice -15 myp&         #以 15 为优先数运行程序 myp
#nice -n-15 myp&       #以-15 为优先数运行程序 myp
#ps -l | grep myp      #查询进程 myp 的运行级和 NICE 值
#输出中的 S 栏为进程状态，PRI 栏为优先级，NI 为 NICE 值。
```

6．改变正在运行进程的优先级（renice）

nice 的功能是设置将要运行程序的 NICE 值，renice 可以调整一个正在执行进程的 NICE 值，其用法为：

```
renice [-n] priority [-g|-p|-u] identifier...
```

选项-n 是可选的，priority 为新的 NICE 值，-g、-p、-u 分别用于指定进程组、进程和用户（名或 ID）。若指定了进程组或用户，则进程组或用户所有进程的 NICE 值都被调整。

6.5.3　改变进程的家目录（chroot）

1）功能、用法及说明

chroot（change root）的作用是改变进程的根目录。其用法为：

```
chroot    newroot    [cmd]
```

参数 newroot 为 chroot 改变以后进程使用的新“根目录”，可以使用绝对路径或相对路径。cmd 为以 newroot 为新“根”文件系统内存放的命令，如果不提供 cmd，则默认使用环境变量 SHELL 指定的 shell。如果指定了 cmd，则 cmd 不得是一个通过符号链接指向 newroot 外部的文件，如果是指向 newroot 内部的，则应以相对路径方式链接，因为如果此时 chroot 能够成功，则 cmd 已经不能再看到 newroot 以外的整个文件系统了，从而使得符号链接失效，同样 cmd 所需要的其他相关文件也必须相对于新的根存在。

若要成功执行 chroot，则必须事前有一定的准备。首先，新的根目录 newroot 必须存在；其次，命令 cmd 必须事先在 newroot 目录内的合适位置存在，且最好是采用静态链接生成的，以备成功改变根目录后以 newroot 为根目录时能够找到它，并能够独立执行，否则执行时所需要的动态库也必须出现在新根目录内的合适位置。

为了找到 cmd 所用的共享库，可以使用 ldd cmd 命令查询，然后按照查询结果创建相应的目录，并将相关的库和命令复制到 newroot 内对应的目录内。

2）准备

以

```
chroot   newroot   cmd
```

为例说明 chroot 命令的准备和使用过程：

（1）查询 newroot 目录是否存在，若不存在则先创建之。

（2）查询 cmd（如$SHELL）的位置，并记住该位置。

（3）查询 cmd 所使用共享库，并记住库名和位置。

（4）在 newroot 下创建命令和共享库所用的每个目录。

（5）将命令、共享库文件和其他必需的文件复制或（硬）链接到相应位置。

（6）执行命令 chroot newroot cmd。

（7）若成功，则会在 cmd 结束时退出 chroot。对于交互式 shell 型程序，可输入^D 或 exit 后回车退出。

第（1）～（5）步是真正的准备过程，而（6）是对准备工作的检验，（7）是 chroot 的退出或返回。

3）chroot 示例

这里仅以在 32 位 Fedora 24 下，将目录切换到当前目录下的 nroot，默认要执行的是环境变量$SHELL 所指定的命令，即

```
chroot   nroot   $SHELL
```

为例说明准备过程，大致过程如下。

```
#chroot nroot                    #执行 chroot nroot，得出以下错误，说明目录不存在
##chroot: cannot change root directory to nroot: No such file or directory
#mkdir -p nroot                  #创建目录 nroot
#chroot nroot                    #再次执行 chroot nroot，得出以下错误
##chroot: cannot run command `/bin/bash': No such file or directory
##说明/bin/bash 不存在。创建目录 nroot/bin 并复制文件/bin/bash 到其中
#mkdir -p nroot/bin;             #创建目录 nroot/bin
#cp $SHELL nroot/bin             #将/bin/bash 复制到/nroot/bin
#chroot nroot                    #再次执行 chroot nroot，仍得出以下错误
##chroot: cannot run command `/bin/bash': No such file or directory
##检查/bin/bash 的支持库
#ldd $SHELL                      #检查/bin/bash 依赖库文件，得出以下输出
        linux-gate.so.1 (0xb77c8000)
        libtinfo.so.6 => /lib/libtinfo.so.6 (0xb7679000)
        libdl.so.2 => /lib/libdl.so.2 (0xb7674000)
        libc.so.6 => /lib/libc.so.6 (0xb74a6000)
        /lib/ld-linux.so.2 (0x8002a000)
##创建目录 nroot/lib 并将以上依赖文件复制到其中
#mkdir -p nroot/lib              #创建所需目录
#cp /lib/libtinfo.so.6 nroot/lib #依次复制相关文件
#cp /lib/libdl.so.2 nroot/lib
#cp /lib/libc.so.6 nroot/lib
```

```
#cp /lib/ld-linux.so.2 nroot/lib
#chroot nroot                     #执行 chroot nroot 成功，输出以下提示符
##bash-3.2#
```
到此为止，准备工作完毕，下边执行几个命令以观察效果。
```
#pwd                              #执行内部命令 pwd，成功，得到输出为/
#echo $SHELL                      #执行内部命令 echo，成功
#ls                               #执行外部命令 ls，失败，得出以下错误提示
##bash: ls: command not found
```

因为 ls 为外部命令，没有为它准备，故在以 nroot 为根目录的新位置文件找不到它，因此不能执行。为了让/bin/ls 能够执行，还需要继续准备，过程同前。

由上可见，为 chroot 做准备工作并不太容易，不过就一般的应用来讲，所需文件并不太多。

对其他命令的准备，或在其他系统下对命令的准备与此过程大致相同，只需参照处理就可以。我们将在第 11 章的 shell 编程中给出一个用于 chroot 准备的通用 shell 脚本程序。

6.6 任务的自动调度

在 UNIX/Linux 系统中，除使用相关命令可对正在运行的程序进行管理和控制外，还有相关命令用于对作业进行调度，以规定在指定时刻调度或安排任务的执行。这样做是非常有益的，因为我们可以把那些不急于处理的任务，放在系统负载小或非峰值的时间执行，也可以让系统在无用户使用或用户最少的时候重新启动，以保证系统运行的效率。用于作业管理的命令有 at、batch 和 crontab 等。

6.6.1 at 和 batch

1．功能与用法

at 和 batch 的功能是在某指定时刻调度任务的执行，其用法为：

```
at [-V] [-q queue] [-f file] [-mMlv] <timespec... | -t time>
at -c job [job...]
atq [-V] [-q queue]
at [-rd] job [job...]
atrm [-V] job [job...]
batch
```

at 和 batch 为作业提交的客户端程序，由 at 和 batch 提交的作业由 atd 调度执行。

2．参数说明

at 的部分参数如表 6-12 所示。

表 6-12　at 的部分参数

选　　项	功 能 描 述
-c job	将指定作业信息显示到标准输出
-d|-r job	删除指定作业
-l	列出队列中的作业
-f file	从文件 file 中读取作业，而非从标准输入，也可以用输入重定向的办法实现
-m	作业完成后向用户发送邮件，在作业没有输出和错误时也是这样
-q queue	指定作业队列 queue。用单字符表示作业队列，可使用的有 a~z，A~Z。a：at 队列；b：batch 队列
-t time	用于指定[[CC]YY]MMDDhhmm[.ss]格式的时间

at 从标准输入或-f 指定的文件读取任务，也可以用输入重定向的办法实现。

at 可以接收的 timespec 时间格式为 HH:MM 或 HHMM 形式的时间，默认日期为当天，若当天时间已过，则为下一天，时间还可以跟后缀 AM 和 PM，否则将视为 24 小时制。用户也可设定 today、tomorrow、midnight、noon、teatime（4pm）。还可以同时指定命令执行的时间和日期，日期必须跟在时间的后面，其格式可以是：（1）英文月名 数字日期 [数字年]；（2）MMDDYY；（3）MM/DD/YY；（4）DD.MM.YY。

用户也可以“当前时间+偏移量”的办法来指定执行时间，这里的偏移量可以是分钟、小时、天或周等。例如，now + 2 minutes，10:22 + 3 days 等。

如果在作业操作中未使用 I/O 重定向，则 at 在作业执行过程中产生的标准错误和标准输出将以邮件的形式发送给用户。

atq 用于查询-q 指定的作业队列，atrm 用于删除作业。

batch 是一个 shell 编写的程序，它通过 at 提交批处理作业。batch 不接收命令行参数，但可通过输入重定向从脚本文件向它提交任务。batch 最终执行的命令为：

```
at -qb now
```

由此可见，batch 所使用的队列为 b（at 所使用的队列为 a）。如果系统的平均负载不超过 atd 规定的上限（默认为 0.8），batch 将立即调度作业执行，否则另择时执行。

超级用户可以无条件使用 at，对于一般用户能否使用 at 要根据文件/etc/at.allow 和/etc/at.deny 来确定。/etc/at.allow、/etc/at.deny 文件的结构为每行一个用户列表，其中不允许出现空格。如果/etc/at.allow 存在，则其中的用户将被允许使用 at。若/etc/at.allow 不存在，则检查/etc/at.deny，其中的用户将不允许使用 at。空的/etc/at.deny 意为每个用户均可使用 at。如果/etc/at.allow 和/etc/at.deny 均不存在，则只有超级用户能使用 at 命令。与 at 相关的文件还有/var/spool/at 和/var/spool/at/spool。

3．作业使用示例

1）作业创建

```
#at 1200 < myjob                #通过输入改道从 myjob 中读取作业，并规定在 12:00 执行
#at 12:10 -f myjob              #从 myjob 中读取作业，并规定在 12:10 执行
#at 1230                        #标准输入读取作业，并在 12:30 执行
at>ls -l >/tmp/`whoami`.a       #进入 at 界面，输入相关命令。“at>”为提示符
```

```
at>date >>/tmp/`whoami`.a            #继续输入命令
at>Ctrl_D                            #按 Ctrl_D 结束作业输入
at><EOF>                             #at 显示此行内容，并覆盖 Ctrl_D 所在的行
```

2）列出已提交的作业

```
#at -l
```

3）删除已提交的作业

```
#at -d 23 25                         #23，25 为 at -l 输出中的作业编号
```

6.6.2 crontab

1．功能与用法

crontab 的功能是规划程序周期性地定时启动、定时执行。crontab 是一个客户端程序，用于编辑用户 crontab 文件。crontab 提交的任务由服务器程序 crond 调度。

at 和 batch 提交的作业是一次性的，而 crontab 提交的作业是重复性的、永久性的，当 crontab 被制定好之后，在下一次被修改之前 crond 将一直按其内容的规定调度程序的执行。crontab 的用法为：

```
crontab [-u user] file
crontab [-u user] [-l | -r | -e] [-i] [-s]
```

crontab 受文件/etc/cron.allow 和/etc/cron.deny 的控制。如果/etc/cron.allow 存在，则只有列在其中的用户才能使用 crontab。如果/etc/cron.allow 不存在，而/etc/cron.deny 存在，则只有不在/etc/cron.deny 中的用户才能使用之。若两者均不存在，则根据系统配置，可能只有超级用户可使用 crontab 或所有用户均可使用 crontab。在 Linux 中，默认允许所有用户使用 crontab。

2．参数说明

crontab 命令的参数如表 6-13 所示。

表 6-13　crontab 命令的参数

选　　项	功 能 描 述	选　　项	功 能 描 述
-e	编辑用户的 crontab 文件	-u user	指定用户的 crontab 文件
-l	列出用户 crontab 文件内容	file	指定 crontab 文件。默认用户自己
-r	删除用户 crontab 文件	-i	与-r 配合使用，删除用户 crontab 文件进行提示

3．crontab 文件的内容

系统将为每个可以使用此功能的用户，以用户名在/var/spool/cron 下创建与用户名相同的 crontab 文件，如 root、test 等。

用户的 crontab 文件是由 crontab 命令来修改的，当使用-e 修改用户的 crontab 文件时，crontab 命令自动调用 vi 对文件进行编辑，当从 vi 中退出时，将修改后的内容回写，任务自

动提交（说明：ubuntu 允许在首次使用 crontab -e 时从已经安装的编辑器 ed、nano 或 vi 中选择一个编辑器，不妨选择 vi）。

crontab 文件中，#开始的行为注释行、空行无效，其他为有效行。除了用于变量定义的一些特殊行外，每一个有效行由 6 个用空格分隔的域组成，格式如下：

```
minute hour day    month day_of_week    command
  分     时    月      月         周日期            命令
```

前面 5 个是数字，分别是分钟（0-59）、时（0-23）、日期（1-31）、月（1-12）和星期几（0-6，0 表示星期天），月份和星期几也可以用英文表示。它们可以是以下形式：

（1）数字。

（2）由减号连接的两个数字，表示范围，如 1-5 表示从 1 到 5。

（3）一组由逗号分隔的数字（枚举），如 1，3，5。

（4）*表示所有或任何允许的值。

（5）*/s 表示步长，比如，0-23/2 表示从 0 到 23 每隔 2 个单位发生一次，即 0，2，4，…

若在文件中同时指定了两个日期，即日期和星期，则两者都有效；如果只想指定一个日期，则只需指定其中一个，而另一个用*表示。

第 6 个域是将要在指定时间执行的命令字符串。命令串中的%（\%除外）将被解释为换行符，%后的内容将作为命令的标准输入。

在 crontab 文件中也可定义供 crond 使用的环境变量。一些变量是由 crond 自动设置的，如 SHELL、HMOE 和 LOGNAME 等。它们中的一些是可以被重新定义的，但 LOGNAME 不能。用户也可定义或修改变量，比如，MAILTO 变量定义在 crond 执行时发送给邮件的用户。定义方法与 shell 编程相同，比如：

```
MAILTO=gjshao
```

4. crontab 示例

1）crontab 文件示例

```
##定义环境变量 SHELL
SHELL=/bin/sh
##定义邮件接收者
MAILTO=gjshao
##每天的 00:05 执行$HOME/bin/daily.job（假设的任务和程序）
5 0 * * *          $HOME/bin/daily.job >> /tmp/out 2>&1
##每周日 3:30 重启系统
30 3 * * 0         /sbin/init 6
##从周一～周五每天 22:00 向 gjshao 发邮件
0 22 * * 1-5       mail -s "It's 10pm" gjshao%Hi,%%Where are your?%
##在偶数点的半点向系统中的用户广播信息
30 0-23/2 * * * wall "run 30 minutes after midnight, evry 2 hours,everyday"
```

2）编辑自己的 crontab 文件

```
#crontab -e
##进入编辑界面，输入以下内容    ##每周五 3:30 重启系统
30 3 * * 5        /sbin/init 6
##每周日的 0:30 清除/tmp/gjshao*文件
30 0 * * 0        rm -f /tmp/gjshao* &> /dev/null
```

3）查看 crontab 文件的内容

```
#crontab -l
```

6.6.3　系统 crontab

使用 crontab -e 编辑提交的 crontab 任务是用户级的，存储在/var/spool/cron/[crontabs/]user 内。相对应的，还有系统级的 crontab，可存储在/etc/crontab 和/etc/cron.d/*中。/etc/cron.d/内是一个个文件可以用来按用户、属性或用途分类存储任务。系统级 crontab 文件结构如下：

```
minute hour day month day_of_week    username    command
  分     时    日     月        周日期          用户名        命令
```

比用户级的 crontab 文件多了一个 username 项以区别任务所有者。

系统级的 crontab 文件只有 root 才能编辑修改。若要向系统级的 crontab 文件中添加任务，只需要按照格式要求输入正确的内容并保存就可以了，不需要重启 cron 服务。

6.7　进程管理图形界面

用户可通过 Applications→Utilities→System Monitor 或在 CLI 下执行 gnome-system-monitor 命令打开资源管理器，选择进程资源状态图后右击所需进程，在弹出的菜单中选择功能对进程进行操作。可进行的操作有 Stop（暂停）、Continue（继续）、End（结束）、Kill（终止）、Change Priority（改变优先级）、Memory Maps（查看内存映像）、Open Files（查看进程打开的文件）和 Properties（查看进程属性）。

习题 6

1．思考题

（1）UNIX/Linux 系统有几类进程？试说明后台进程的作用或执行过程。
（2）试述 0#和 1#进程的作用及 UNIX/Linux 进程树的形成过程。
（3）试述 UNIX/Linux 系统的运行级和 init、telinit、systemctl 的作用。
（4）Linux 系统中，进程常见状态标志有哪些？各是什么意义？
（5）系统启动管理软件包的发展经历了哪几个阶段？在不同阶段的包名是什么？
（6）登录 shell 是如何启动的？如何定制用户自己的 shell 启动？

2．单项选择题

（1）在 Linux 系统中用于进程状态查询的常用命令是（　　）和（　　）。

A．ps　　B．kill　　C．killall　　D．pstree

（2）若使 pid 进程无条件终止，使用的命令是（　　）。

A．kill -9　　B．kill -15　　C．killall -1　　D．kill -3

（3）当一个进程正在执行时用户按下了 Ctrl_C 键，该进程接收到的信号是（　　）。

A．KILL　　B．2　　C．1　　D．TERM

（4）设置用户作业被定义在文件/tmp/myj 内，要求该作业在 3 小时以后执行，正确的命令是（　　）和（　　）。

A．at now + 3 hours < /tmp/myj　　B．at now + 3 hours /tmp/myj

C．at 3:00 hours < /tmp/myj　　D．at now + 3 hours -f /tmp/myj

（5）在 Linux 系统中，让命令 myp 以优先数 10 运行的命令为（　　）和（　　）。

A．nice -10 myp　　B．nice -n10 myp

C．nice 10 myp　　D．nice +n10 myp

3．综合题

（1）试述 kill 与 killall 的相同与不同点。

（2）试述 fuser 的功能与作用，如何终止在终端 tty2 运行的所有进程？如何监视网络服务端口的活动情况？

（3）简述 UNIX/Linux 的作业和任务调度功能，如何让作业 myjob 在两小时后执行？如何让系统在每周一 1:00 重新启动？

（4）试描述 suid、sgid 和 sticky 机制，回答如何避免它们带来的安全隐患？分别查找系统中具有 suid、sgid 和 sticky 属性的文件。

实验 6

1．进程状态查询与终止指定进程。
2．同时在多个终端登录后，终止所有的同名进程 bash，并观察系统的反映。
3．通过作业和任务管理让系统在今后某个时刻关机或重新启动。
4．使用 rc.local 文件，让系统启动时，将启动的大致时间记录到/tmp/time.start。

第 7 章　系统安装、扩充、启动与管理

7.1　系统安装

每个 Linux 系统都有多种安装模式，也有不同的安装方法，这里只介绍使用发行光盘进行安装的方法。

7.1.1　安装任务与准备

系统安装任务是将发行介质上操作系统映像等信息安装到硬盘上，在系统安装前要进行一定的准备。

1．硬件准备

一般来说，流行或较流行的计算机系统均可安装本书所使用的 Linux 系统，甚至稍落后一点的也能安装并运行它们。当然这样的安装只能供一般学习、家用或办公之用，如果是应用于某些业务系统、商业运营，或用于云计算大数据等服务，则需要更高的硬件支持。

比如，Fedora 25 的最小安装要求是 1GHz CPU，1GB 内存，10GB 硬盘空间，显卡分辨率为 800*600。

2．介质准备

安装前应准备好 Linux 系统所需要的光盘或光盘映像，不同版本发行套件中所含的光盘数量不尽相同，并且这种映像也有不同版本，如 Desktop（桌面）、Server（服务器）、Workstation（工作站），还有 live 版的，可以根据需要选择。

3．确定所需要的系统

一般地，Linux 有 3 种典型安装模式：桌面系统、工作站和服务器，以哪种方式安装应取决于用途。一般来说，除非用于研究或实验，安装的内容宁少勿多。安装内容多了，除占用磁盘空间外，还会影响运行效率。若是家用目的，可以采用桌面方式安装，应配置 X 桌面系统。若是用于服务器，因为操作者多为有经验者或专业人员，建议不配置 X 桌面系统，因为图形桌面的运行会占用大量内存，影响服务器的运行效率。不需要的服务不要安装，多余的服务可能也是安全隐患。

7.1.2　硬盘的物理结构与分区划分

1．主引导扇区与分区表

微机硬盘的第一个物理扇区存放的是主引导扇区（MBR：Master Boot Record），它包含机器的主引导程序和整个硬盘的分区表。MBR 的前半部分（<446 字节）为引导程序，最后

2 字节为分区标识，分区标识前的 64 字节是以 16 字节为单位的 4 个分区表。如果硬盘的 MBR 遭到损坏，则将导致整个硬盘数据无法正常使用。可用特殊的办法或工具对主引导扇区进行操作或维护，但要特别谨慎。

MBR 不属于任何一个分区和操作系统，但它是引导型病毒常感染和修改的地方。

MBR 的大部分内容是所有硬盘可公用的，即与机器或硬盘无关，可以互相替代，但分区表部分是私有的，基本上每个硬盘都不一样，因此 MBR 的损坏最可怕的是分区表的损坏。

主引导扇区不属于任何分区，它的结构如下。

```
struct {
    unsigned char boot_code[446];        //引导代码（可能会因机器而异）
    partition_table partition_table1;    //从 1BEH 开始，大小 16 字节
    partition_table partition_table2;    //从 1CEH 开始，大小 16 字节
    partition_table partition_table3;    //从 1DEH 开始，大小 16 字节
    partition_table partition_table4;    //从 1EEH 开始，大小 16 字节
    uint8_t partition_mark1;             // 0x55
    uint8_t partition_mark2;             // 0xaa
} MBR
```

每个分区表的结构如下。

```
struct {
    uint8_t status;                 //0 或 0x80，0x80 为活动分区
    uint8_t start_head;             //开始磁头，最大为 255
    uint8_t start_sector_cyl[2];    //6 位用于开始扇区，10 位用于开始柱面
    uint8_t type;                   //0x83-Linux，0x63-UNIX，0x0f-扩展分区等
    uint8_t end_head;
    uint8_t end_sector_cyl;
    uint32_t firstLBA;              //分区开始的绝对扇区号
    uint32_t lengthLBA;             //无符号长整数，分区内扇区总数
} partition_table;
```

2．微机硬盘的结构

微机硬盘的分区格式有两种：MBR 和 GPT。下面简单介绍一下这两种格式。

1）MBR 格式

MBR 格式源自传统的 DOS 系统。根据主引导扇区的结构可知，微机的硬盘最多由四个基本分区组成，这在计算机发展的早期是可以的，但随着技术的发展，不能满足实际需要了。于是就有了扩展分区和逻辑分区的概念。

扩展分区是分区的一种，是 4 个基本分区中的一个，通常是（有效的）物理分区的最后一个，且一般较大，目的是要在扩展分区中创建分区。这种创建在扩展分区中的分区称为逻辑分区。

扩展分区内的逻辑分区按单向链的方法组织（如图 7-1 所示），每一个分区引导扇区的分区表位置只使用前两个分区表项，第一个（myself）是当前逻辑分区的入口，另一个（next）指向下一个分区。若是最后一个逻辑分区，则 next 为空（全为 0）。

```
struct {
      partition_table myself;          // my logical entry
      partition_table next;            // NULL if the last one
} logical_partition;
```

图 7-1 逻辑分区组织的单向链

含有扩展分区的磁盘最大分配情况如图 7-2 所示。

<table>
<tr><td>物理分区 1</td><td></td></tr>
<tr><td>物理分式 2</td><td></td></tr>
<tr><td>物理分区 3</td><td></td></tr>
<tr><td rowspan="4">物理分区 4（扩展分区）</td><td>逻辑分区 1</td></tr>
<tr><td>逻辑分区 2</td></tr>
<tr><td>……</td></tr>
<tr><td>逻辑分区 n</td></tr>
</table>

图 7-2 MBR 格式硬盘结构示意图

在图 7-2 所示的微机硬盘分区结构中，分区的个数应由具体使用情况而定。硬盘上通常会有多个分区，如 D 盘、E 盘和 F 盘等，但它们往往不是基本分区，而是逻辑分区。

在如图 7-2 所示的分区结构中，任何一个分区均可不存在。如果某个基本分区不存在，或基本分区在 4 个以内，则仅前面分区表被使用，后面空分区表填以 0x00。如果逻辑分区中的某个不存在，则只需要调整分区链指针就可以。

由图 7-2 所示的分区表结构的 firstLBA 和 lengthLBA 类型 uint32_t 可知，它们能表示的最大数为 2^32 个扇区（2TB），但现在大容量的存储设备出现了，需要突破这个瓶颈。

2）GPT 格式

GPT（GUID Partition Table）是全局唯一标识（GUID，Globally Unique Identifier）分区表，是一种磁盘分区表结构布局的标准。它是可扩展固件接口（EFI，Extensible Firmware Interface）标准的一部分，用于替代 BIOS 系统中 32 位逻辑块地址和大小的主引导记录（MBR）分区结构。

GPT 使用 64 位逻辑块地址，对于每个扇区大小为 512 字节的磁盘，意味着可以寻址和容纳 2^64 个 215 大小的扇区，按 K、M、G、T、P、E、Z、Y 顺序，最大可达 8ZB。

GPT 使用逻辑块地址 LBA（Logical Block Address）的概念。LBA 将磁盘上的所有存储单位（扇区）从 0 开始编号直到最后一个 n-1，依次为 LBA0，LBA1，…，LBAn-1。

GPT 格式的磁盘结构如图 7-3 所示。

<table>
<tr><td>MBR</td><td>GPT 头</td><td>分区表</td><td colspan="4">分区实体集</td><td>备份分区表</td><td>备份 GPT 头</td></tr>
<tr><td>LBA0</td><td>LBA1</td><td>LBA2~LBA34</td><td>分区 1</td><td>分区 2</td><td>…</td><td>分区 m</td><td>LBAn-33~LBAn-2</td><td>LBAn-1</td></tr>
</table>

图 7-3 GPT 格式的磁盘结构示意图

（1）MBR

出于兼容性考虑，硬盘的第一个扇区（LBA0）仍然用作 MBR，结构也与原来的相同，前半部分（<446 字节）仍为引导代码，但是 4 个分区表只使用了一个，分区类型为 0xEE，以此来表示这块硬盘使用 GPT 分区表。引导代码在机器启动时不再发挥作用。

（2）GPT 头及格式。

LBA1 是 GPT 头，可以理解为硬盘的属性描述块，包括可用空间、分区表项的大小和数量等。GPT 的结构如下所示。

```
#define GPT_RESERVED 420
struct GPTHeader {
    uint64_t signature;                 #GPT 头签名（“EFI PART”）
    uint32_t revision;                  #版本号
    uint32_t headerSize;                #头的大小
    uint32_t headerCRC;                 #CRC 检验和
    uint32_t reserved;                  #保留 1，为 0x00000000
    uint64_t currentLBA;                #GPT 头所在的 LBA 号
    uint64_t backupLBA;                 #备份 GPT 头所在的 LBA 号
    uint64_t firstUsableLBA;            #首个可用的 LBA 号（分区实体区首址）0x22，即 LBA34
    uint64_t lastUsableLBA;             #最后一个可用的 LBA 号（备份分区的首址-1）
    GUIDData diskGUID;                  #磁盘 GUID（全球唯一标识符，与 UUID 同义）
    uint64_t partitionEntriesLBA;       #分区表起始扇区号，通常为 0x02，即 LBA2
    uint32_t numParts;                  #分区表总项数，通常限定为 0x80，即 128 个
    uint32_t sizeOfPartitionEntries;    #每个分区表项占用字节数，通常限定为 0x80，即 128 字节
    uint32_t partitionEntriesCRC;       #分区表 CRC 校验和
    unsigned char reserved2[GPT_RESERVED];    #保留 2，头扇区的剩余部分通常全填充为零
}; // struct GPTHeader
```

（3）分区表项及格式。

在 GPT 头之后的 34（0x22）个扇区，LBA2~LBA33 的内容为分区表项。分区表项的结构如下。

```
#define NAME_SIZE 36 // GPT allows 36 UTF-16LE code units for a name in a 128 byte partition entry
GPTPart {
        PartType partitionType;         #GUID 分区类型
        GUIDData uniqueGUID;            #GUID 分区唯一标示符
        uint64_t firstLBA;              #分区的起始 LBA 编号
        uint64_t lastLBA;               #分区的结束 LBA 编号
        Attributes attributes;          #分区的属性标志
        uint16_t name[NAME_SIZE];       #分区名（最多 36 个小端格式 UTF-16 字符）
} // GPTPart
```

（4）其他。

备份 GPT 头是对 GPT 头的备份，一般位于磁盘的最后一块。

备份分区表是对分区表的备份，按照原来顺序存放在备份 GPT 头前连续的区域内。

分区实体集为与分区表项对应的一个个分区实体的集合，其中可以有各种类型的分区，但分区个数往往会小于分区表项个数，其中也可能会有多个未被分配的磁盘空间。

3．系统引导

在 MBR 结构系统中，只有安装在活动分区（active partition）上的操作系统才是可引导

的。在系统加电后，首先读取 MBR，执行其中的代码，扫描分区表，从中找到活分区，然后读入活分区上操作系统的引导扇区，把控制权交给它，在它的控制下引导操作系统。

在 GPT 格式的 EFI 系统中，系统的引导不再需要 mbr 主引导记录，要依靠 UEFI BIOS 和硬盘上的 EFI 分区，这里不再做进一步讨论。

每个操作系统都有自己的引导扇区（boot sector），称为逻辑引导扇区或逻辑引导记录（LBR，Logical Boot Record）。逻辑引导扇区位于操作系统所占分区的第一扇区。引导扇区的内容因操作系统的不同而不同。有的既包括引导程序又包括文件系统的控制信息（DOS/Windows），有的则只包含引导程序（UNIX/Linux）。逻辑引导扇区的作用是负责本操作系统的引导或提供相关文件系统信息。如果是（活）分区是 Linux，则 Linux 系统就被启动了。

7.1.3 安装 Linux 系统所需的基本分区

在操作系统安装之前，应根据该系统所承担的工作等实际情况和实际需要对系统所使用的硬盘进行规划。对于 Linux 系统独占硬盘的情况可以采取自动分区办法来规划分区，但这样分区的结果不一定是最佳的。

1．引导分区

引导分区是在硬盘上用于系统启动的分区，不需要太大，够用就行。在 Linux 中有 500MB 就足够了。在 MBR 格式系统中，可引导分区入口必须位于硬盘的 1024 道之前。

在 Linux 系统中也可不设置独立的引导区，此时它共享系统的根分区。这种情况下，对于 MBR 结构的系统，根分区必须位于 1024 道之前，或者将 Linux 的引导程序 Grub 安装在 MBR 上。

2．根分区

根分区是 Linux 系统的主要工作分区，不能太小，要根据安装系统的内容和将来的用途来决定，建议 10GB 以上。如果要在 Linux 系统上运行大型数据库则可能会需要更多硬盘空间，当然数据库也可占用单独的裸分区。

3．交换区

交换区是 Linux 系统用于动态扩充内存的，安装时可以不指定，但是为了保证安装的成功率，建议设置独立的交换区。这个用于交换区的独立分区可是硬盘任何形式的分区，大小可参考实际物理内存来确定。对于 8GB 内存以下的系统，建议交换大小为物理内存的 1.5～2.5 倍，或再大一点；对于内存为 8GB 或以上者，可与内存容量相等，或小于物理内存。

不论是否指定了交换区，系统安装成功后，在运行时都是需要交换区的。若未指定交换区，则系统会在自己的根分区上找一块连续的区域用作交换区，此时交换区会占用系统根分区空间。

系统安装完毕后，交换区还可以被扩展或更换，请参见交换区管理部分内容。

4．其他分区

仅从安装的角度，有引导区、根分区和交换区就可以了。在系统工作时可能还需要/var、/home、/tmp 等分区，若不指定则共享系统的根分区。对于用于网络服务的系统，建议/var

使用独立分区，避免因/var 过多占用，而导致根分区空间被挤占、被用完，从而影响系统的正常运行。同样，对于运行大型数据库的系统，数据库系统所占用的分区也要足够大，或使用独立的磁盘或磁盘分区。

7.1.4　安装过程

Linux 系统的安装是一个漫长的过程，根据安装内容的多少，可能需要几十分钟。不同系统的安装过程有差异。Fedora 24 的安装过程大致如下。

（1）放入启动光盘，加电启动。

加电之后，系统开始启动，很快会出现如图 7-4 所示的界面。

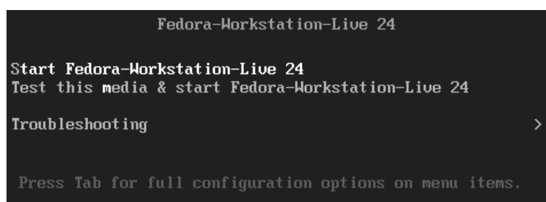


图 7-4　安装方式选择界面

通过“↑”或“↓”选择第一项“Start Fedora-Workstation-Live 24”，按回车键进行“Live”系统启动。

在如图 7-4 所示界面中有个 1 分钟的超时时限，若在时限内没有键盘输入动作或者选择了“Test this media & start …”，则先进行光盘介质检测，此时可以按 ESC 键中断检测过程继续进行“Live”系统启动。

（2）选择“Install to Hard Drive”。

“Live”系统启动完成后会出现一个用户可以选择“Try Fedora”（体验 Fedora），或“Install to Hard Disk”（将 Fedora 24 安装到硬盘上）的界面（图略）。

选择“Install to Hard Disk”进行系统安装，之后进入安装语言选择界面（图略）。

（3）选择安装语言。

用户可以根据需要选择安装语言，如“English→United States”或“中文→简体中文”等，这里选择“English”。

选好安装语言后，单击“Next（前进）”按钮，进入如图 7-5 所示的“INSTALLATION SUMMARY（安装信息摘要）”界面。

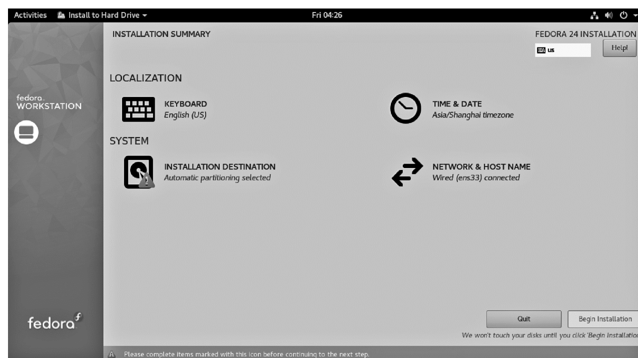


图 7-5　安装信息摘要界面

在如图 7-5 所示界面中可以设置的有 LOCALIZATION（本地化）和 SYSTEM（系统）两大类。

（4）本地化设置。

在本地化设置部分有“KEYBOARD（键盘）”和“TIME & DATE（时间和日期）”设置。用键盘选择“English (US)”；时区设为“Asia/Shanghai”，时间和日期设置正确，或安装后再设置。

（5）系统设置。

在 SYSTEM（系统）设置部分有“INSTALLATION DESTINATION（安装位置）”和“NETWORK & HOSTNAME（网络和主机名）”。网络和主机名可先不设置，等到安装完成后根据需要再设置。

（6）设置安装目标。

本部分用于设置安装目标磁盘上的用于 Linux 的分区。由图 7-5 所示界面可以看出，在“INSTALLATION DESTINATION（安装目标）”处有一个红色警告标志，并在界面底端有一行黄色提示“请先完成带有此图标标记的内容再进行下一步”，也就说此部分必须设置正确。单击“INSTALLATION DESTINATION（安装位置）”进入如图 7-6 所示的安装位置设置界面。首先勾选所用硬盘。

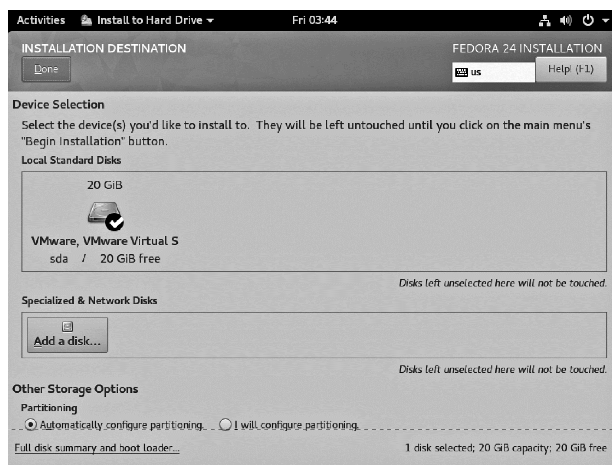


图 7-6　安装位置设置界面

在此界面里有两个选项，“Automatically configure partion（自动分区）”和“I will configure partion（手动分区）”，还有一个“Full disk summary and boot loader（完整磁盘摘要及引导程序）”的按钮，用于选择安装引导器（boot loader）的设备，若只有一个硬盘那就别无选择了，若有多个则可以根据实际需要进行选择，若不安装引导器，则安装后无法启动。

如果磁盘有足够多的剩余空间或将整个硬盘都分配给 Linux，则可选择自动分区。只需在勾选安装硬盘后单击“Done（完成）”返回就可以。

若选择“手动分区”，需要有足够的经验。假设整个硬盘是空的或有足够剩余空间（如 20GB）。选择“手动分区”后，单击“Done”按钮进入如图 7-7 所示的手动分区设置界面。

在如图 7-7 所示界面上，依次单击“+”，参考如图 7-8 所示界面，分别创建 boot 分区（/boot）、交换分区（swap）和根分区（/），它们的大小分别依次为 500MB、2GB 和 17.57GB，创建后分区的展示界面如图 7-9 所示。如果发现某个分区创建得不合适，则可单击“-”删除

后重建。根据需要，还可以创建/var、/tmp 等。对创建工作满意后，单击“Done”，进入如图 7-10 所示的“SUMMARY OF CHANGES（更改摘要）”界面。

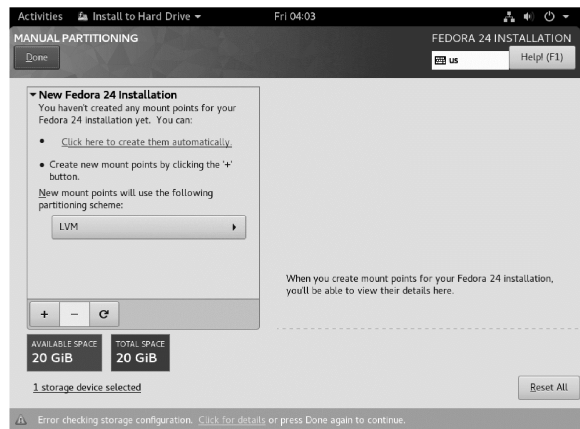


图 7-7　手动分区设置界面

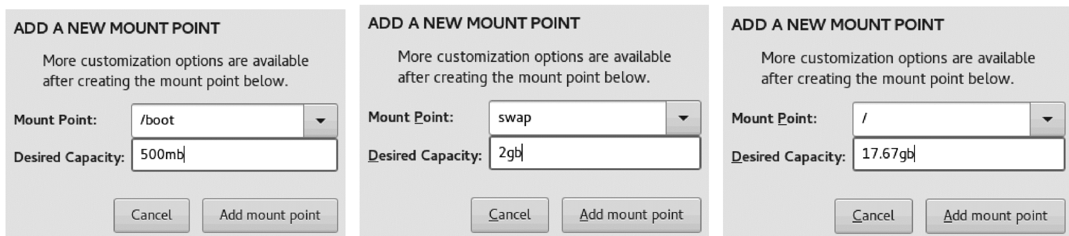


图 7-8　分区创建示例

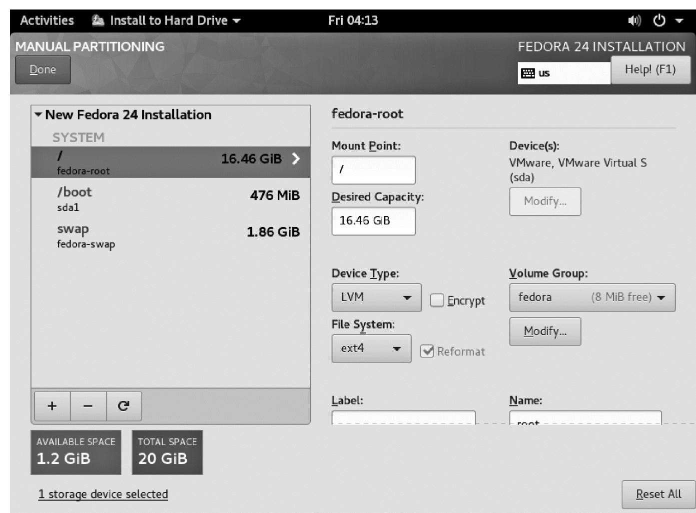


图 7-9　创建后分区的展示界面

在分区创建过程中，用户可能已经发现，安装程序并不是百分之百按照用户指定的大小去创建分区，而是根据分区边界和磁盘物理情况有所调整，尤其最后剩余的部分，很难正好把它用尽。若欲正好用尽剩余部分，则可以指定分区大小值超过实际剩余值，然后由安装程

序据实创建。

如果是在 EFI 系统中的 GPT 格式磁盘上安装，还需要一个 300MB 左右的 biosboot 分区。这个分区也被称为 efi 分区，它是硬盘上所有操作系统所共享的，若已经存在则不需要再创建。

SUMMARY OF CHANGES

Your customizations will result in the following changes taking effect after you return to the main menu and begin installation:

Order	Action	Type	Device	Mount point
1	Destroy Format	Unknown	VMware, VMware Virtual S (sda)	
2	Create Format	partition table (MSDOS)	VMware, VMware Virtual S (sda)	
3	Create Device	partition	sda1 on VMware, VMware Virtual S	
4	Create Format	ext4	sda1 on VMware, VMware Virtual S	/boot
5	Create Device	partition	sda2 on VMware, VMware Virtual S	
6	Create Format	physical volume (LVM)	sda2 on VMware, VMware Virtual S	
7	Create Device	lvmvg	fedora	
8	Create Device	lvmlv	fedora-swap	
9	Create Format	swap	fedora-swap	
10	Create Device	lvmlv	fedora-root	
11	Create Format	ext4	fedora-root	/

Cancel & Return to Custom Partitioning　Accept Changes

图 7-10　更改摘要界面

在如图 7-10 所示界面中有“Cancel & Return to Custom Partitioning（取消并返回到自定义分区）”和“Accept Changes（接受更改）”供选择。若选择取消，则退回并重新设置，这里选择接受。单击“Accept Changes（接受更改）”，重新回到如图 7-5 所示的“INSTALLATION SUMMARY（安装信息摘要）”界面，所不同的是，如果设置正确，则之前的红色提示不见，且右下角的“Begin Installation（开始安装）”变得可用了。

（7）开始安装。

截止到现在，所有的配置操作还都没有生效，也就是说还都没有写到磁盘上。单击“Begin Installation（开始安装）”开始安装，出现如图 7-11 所示的安装过程界面。

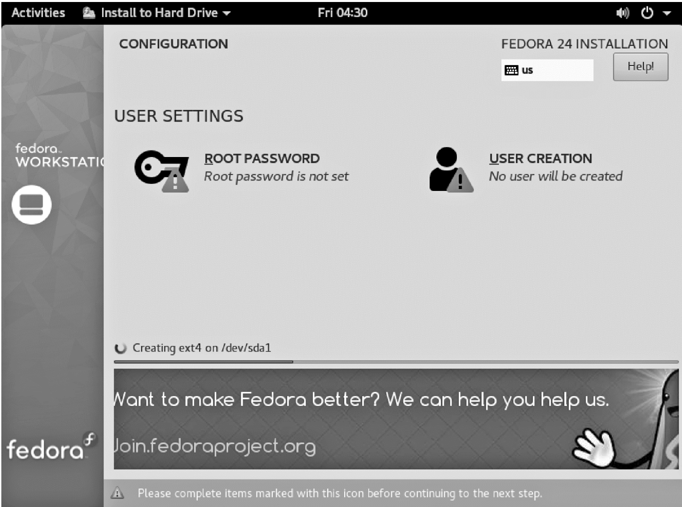


图 7-11　安装过程界面

（8）为 root 用户设置密码和创建普通用户。

在如图 7-11 所示的安装过程界面上，有“ROOT PASSWORD（root 密码）”和“USER CREATION（创建用户）”两个按钮。前者用于为 root 设置密码，必须在安装过程中完成；后者用于创建普通用户，可在安装过程创建，也可在安装完成后首次启动时创建。

单击“ROOT PASSWORD（root 密码）”为 root 用户设置密码；还可以单击“USER

CREATION（创建用户）”创建普通用户。如图 7-12 所示为创建 test 用户并设置密码的界面。在密码和用户设置完成后单击“Done”继续，等待系统安装完成。

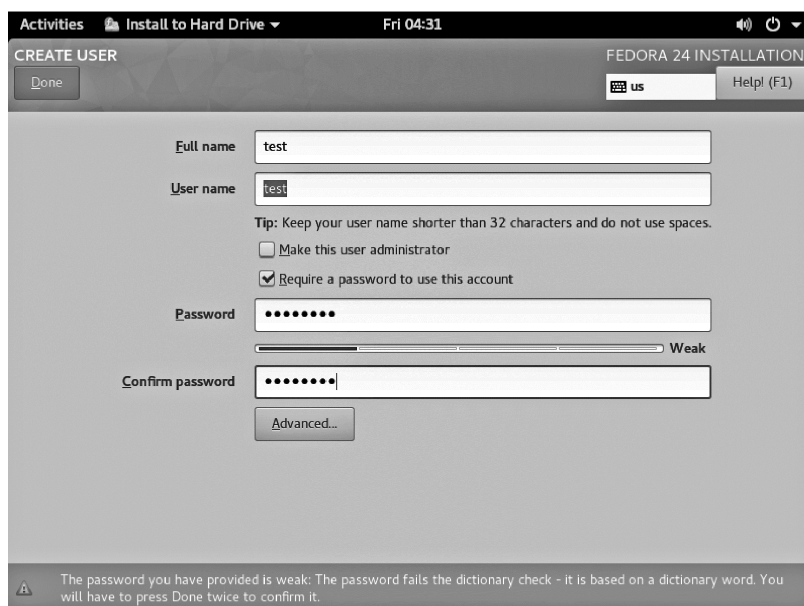


图 7-12　创建 test 用户并设置密码界面

（9）退出安装，重启系统。

一切可做的都做完了，剩下的只有等待。安装完毕后单击“Quit（退出）”退出安装，然后重启系统就可以。

单击屏幕右上角的“ ”，在下拉菜单中单击“ ”，在弹出的菜单中选择“Power Off”关闭系统或选择“Restart”重启系统。

（10）首次启动。

再次开机或启动系统后开始首次启动。

若在安装过程中没有创建普通用户，则会在启动接近完成时，登录界面出现之前，要求创建普通用户。创建完毕之后就会出现登录界面。

选择指定用户（如 root）登录后，会首先出现“Welcome（欢迎）”界面，单击“Next（前进）”进入“Typing（输入）”设置界面，在这里用户可以选择键盘布局或其他输入方式，单击“Next”进入“Privacy（隐私）”设置界面，单击“Next”进入“Online Accounts（在线用户）”设置界面，单击“Skip（跳过）”进入“Ready to Go（准备好了）”提示界面，单击“Start using Fedora（开始使用）”弹出“Getting Started（准备好了）”的一个帮助界面，可以在此获得帮助，或关闭此界面后使用系统。

7.1.5　虚拟机的安装与使用

现在有多款虚拟机（Virtual Machine）软件，可以使用户在一台机器上同时运行多个操作系统，如 Windows、Linux、FreeBSD、Solaris 和 MAC OS 系统等。与多启动系统相比，有更多优势，可使多个操作系统在主系统的平台上，就像标准应用程序那样运行。在虚拟环境

下，每个操作系统都可以进行虚拟分区、配置而不影响真实硬盘的数据。虚拟化环境下可以通过网卡将数台虚拟机连接为一个局域网，甚至可虚拟出比实际硬件还多得多的网卡、CPU等设备，能使 IT 人士在一台硬件系统下多个操作系统间进行开发、调试和测试等。

常用的虚拟机软件有 VMware Workstation、VirtualBox 和 Virtual PC 等，在 Linux 下还有 xen 和 kvm 等。

本书编写过程中系统安装、命令执行与测试、服务器安装、配置与管理等都是在虚拟环境下完成的，用户可以选择某种虚拟机软件供学习和研究使用。

在虚拟机下，Linux 系统的安装与独立系统的安装基本相同，只不过在虚拟机系统的控制下 Linux 系统安装会更容易、安全（不会破坏硬盘上的数据）。在虚拟机管理下，Linux 系统可以从光盘安装，也可从映像文件安装，甚至可以通过网络安装，在系统运行过程中也可以改变光驱，添加或删除设备。

7.2 引导器（grub）

grub 是 Linux 系统的引导器，它是系统启动运行的第一个软件。在 MBR 格式的系统中，grub 可以被安装在系统的主引导扇区或 Linux 的逻辑引导扇区上，这取决于系统安装时的选择。grub 识别多数文件系统和引导程序格式，因此在多个操作系统共享硬盘时，可用它引导不同的操作系统。

现在的 Linux 系统使用的引导器是 grub2，本书只简介 grub2 的相关知识及应用。为了描述方便，不再强调版本。

7.2.1 grub 常用术语

1．磁盘设备及命名

在 grub 中，所有硬盘（不论是 IDE 或 SCSI）均以 hd 开头，命名规则为（hdx）或（hdx,y），x（x≥0）为硬盘设备编号，y（y≥1）为硬盘上的分区编号。例如，（hd0）、（hd1）分别代表第 1 和第 2 块硬盘，（hd0,1）和（hd1,2）分别代表第 1 块硬盘上的第 1 块分区和第 2 块硬盘上的第 2 分区。需要说明的是，在 grub 世界里，硬盘扩展分区的逻辑分区也总是从 5 开始编号。

在现在的 Linux 系统中，分区编号方法与此相同，基本分区从 1 开始编号，扩展分区内的逻辑分区总是从 5 开始编号。

软盘以 fd 开头，命名规则为（fdn），n（n=0,1）为软驱编号。

2．文件名

grub 的文件命名方式与系统一致，但绝对路径不是基于文件系统根（/），而是基于设备的。比如，设备（hd0,2）上的文件系统中有文件/boot/grub2/grub.conf，在 grub 里其文件名为(hd0,2)/boot/grub2/grub.conf。

7.2.2　操作界面

grub 提供有菜单和命令行两种操作界面（见图 7-13 和 7-14）。

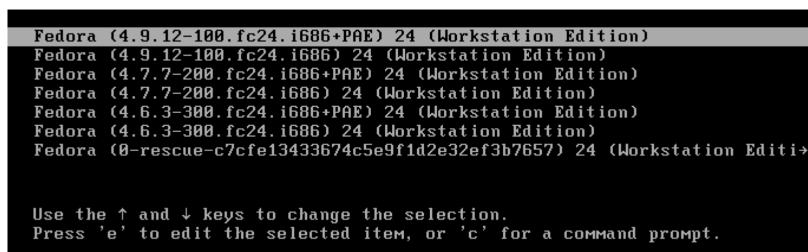


图 7-13　grub 菜单界面

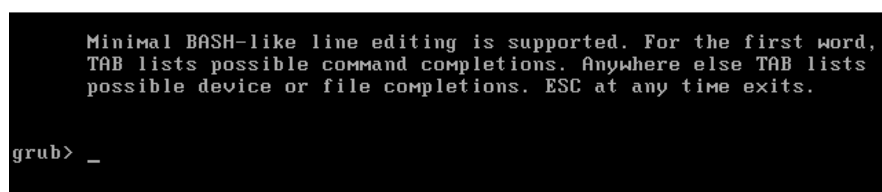


图 7-14　grub 命令行界面

1．菜单界面

系统启动时，grub 的菜单界面如图 7-13 所示，在此界面下，用户可以等待系统自动启动默认启动项，也可以通过“↓”或“↑”键来选择菜单中的其他启动项。当启动项选定之后，可以进行以下操作：

（1）按回车键启动选定项。

（2）按“e”键编辑启动项。

（3）按“c”键进入命令行界面。

用户按“e”键后进入启动项编辑界面，至于说在编辑界面下干什么，根据需要而定。在编辑过程中，若要作废编辑内容则可按“ESC”键返回上级界面；在编辑完成后可按^X 或 F10 键，按编辑后的内容启动系统（示例见后述）。

2．命令行界面

在菜单界面中按“c”键，则 grub 进入命令行界面。进入命令行界面后可以使用 help 命令得到帮助、得到命令列表（也可通过文件/usr/lib/grub/i386-pc/command.lst 得到 grub 的全部命令列表），还可通过 help cmp 得到命令 cmd 的简单帮助。对于 grub 命令的使用需要更多经验，也超出本书范围，不再介绍。但命令中的 halt 和 reboot 可用于关机和重启动。用户也可在命令界面下按 ESC 键返回菜单界面。

7.2.3 grub 及其配置

1. grub2 的新特点

（1）配置文件 grub.cfg 的内容是一个脚本文件，支持 shell 编程。

（2）save_env 和 load_env 支持对启动状态的本地保存。

（3）拥有更加聪明的算法来找寻所需要的文件；使用 search 命令可以通过卷标或 UUID 来选择磁盘或分区。

（4）支持多种类型的系统，如 PC EFI、PC coreboot、PowerPC、SPARC 和 MIPS 等。

（5）支持多种文件系统，不仅仅限于 vfat、ntfs、ext+、hfs+和 iso9660 等。

（6）可以直接从 LVM 和 RAID 中读取文件。

（7）支持图形终端和菜单系统。

（8）支持模块动态加载，可用于支持更多的文件系统和其他功能。

2. 配置目录、配置文件及生成

1）配置文件

grub 的配置文件为 grub.cfg。

2）配置目录

（1）ubuntu 系统 grub 的配置目录为/boot/grub。

（2）红帽系列。

在 MBR 格式的系统中，配置文件为/etc/grub2.cfg 和/boot/grub2/grub.cfg，前者是后者的符号链接。

在 GPT 格式的系统中，配置文件为/etc/grub2-efi.cfg 和/boot/efi/EFI/$OS/grub.cfg，前者是后者的符号链接。此处的 OS 是一个变量，在 Fedora 下为 fedora，在 CentOS 中为 centos，在 RHEL 中为 redhat。

如图 7-15 所示，为获得 grub 配置目录的通用脚本程序 get_grb_cfg_dir 可用于本书所用的所有系统。

```
dmesg | grep "Linux version" | grep "buntu" &>/dev/null
if [ $? -eq 0 ]; then echo "/boot/grub"; exit 0; fi                    #ubuntu
#Red Hat Seriales
rpm -q grub2-efi &>/dev/null
if [ $? -ne 0 ]; then echo '/boot/grub2'; exit 0; fi
dirname `rpm -ql grub2-efi | grep '/boot/efi/EFI/.*/grub.cfg' `
```

图 7-15　get_grb_cfg_dir 脚本程序

为了描述方便，将 grub 的配置文件所在目录统一记为 GRB_CFG_DIR，此值可通过脚本程序 get_grb_cfg_dir 获得，方法是：

```
#GRB_CFG_DIR=`get_grb_cfg_dir`          #获取 grub 配置文件所在目录
#export GRB_CFG_DIR                     #设为环境变量（可选）
```

生成或重新生成 grub 配置文件的方法为：

```
#GRB_CFG_DIR=`get_grb_cfg_dir`          #只需一次
#cd $GRB_CFG_DIR                        #进入配置目录（必须）
#grub2-mkconfig -o grub.cfg             #红帽系统通用
#grub-mkconfig -o grub.cfg              #ubuntu 系统
```

在本书以后的描述中，说生成或重新生成 grub 的配置文件指的就是以上过程。

3．配置目录内的其他文件

在 grub 的配置目录中，除了配置文件 grub.cfg 外，还有许多形如*.mod 的文件，这些是可以动态加载的模块，可在启动时按需要加载它们以实现不同的功能。还有一些类似*.lst 的文件，为可用或所支持的项目列表，如 command.lst 和 fs.lst 分别为命令和文件系统列表；*.img 为 grub 安装或启动时所需要的映像文件；device.map 用于实现 grub 设备到操作系统设备的映射，比如：

```
(hd0)    /dev/sda                       #将(hd0)映射为/dev/sda
(hd0,2)    /dev/sda2                    #将(hd0,2)映射为/dev/sda2
```

4．/etc/default/grub

/etc/default/grub 文件中包含了如默认启动项，超时时限等默认配置；还可设置内核配置行后面的追加参数等。此文件内容将会在 grub 更新配置文件时合并至配置文件 grub.cfg 中。某系统的/etc/default/grub 文件内容如下。

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
GRUB_CMDLINE_LINUX="rd.lvm.lv=fedora_bogon/root rd.lvm.lv=fedora_bogon/swap rhgb "
GRUB_DISABLE_RECOVERY="true"
```

在/etc/default/grub 可以使用的变量如表 7-1 所示。

表 7-1　grub 的常用变量

变　　量	意义或功能	默　认　值
GRUB_DEFAULT	指定默认启动项编号（编号从 0 开始）。当值为 saved 时有特殊含义	0
GRUB_SAVEDEFAULT	如果此选项值为“true”，且 GRUB_DEFAULT=“saved”时，当手动进入一个菜单项后，下次启动时将把它设为默认启动项	false
GRUB_TIMEOUT	设置超时时限，若为−1 则永不启用超时	无
GRUB_HIDDEN_TIMEOUT	设置在显示菜单之前等待键盘输入的时间。如果指定时间内没有键盘操作，那么自动加载 GRUB_DEFAULT 变量所指定的菜单项	无
GRUB_HIDDEN_TIMEOUT_QUIET	与 GRUB_HIDDEN_TIMEOUT 连用，如果设置为“true”，则不显示倒计时	无

（续表）

变　　量	意义或功能	默 认 值
GRUB_DISABLE_RECOVERY	默认情况下，为每个 Linux 内核产生两个菜单项，一个是普通模式，另一个是救援模式。如果此选项值为“true”，则不自动产生救援模式菜单	false
GRUB_CMDLINE_LINUX	指定为 Linux 内核添加的命令行参数	可变
GRUB_CMDLINE_LINUX_DEFAULT	指定为 Linux 内核添加的默认命令行参数。在普通模式下会在 Linux 内核参数后跟上 GRUB_CMDLINE_LINUX 的值，然后再跟上 GRUB_CMDLINE_LINUX_DEFAULT 的值	无
GRUB_DISABLE_OS_PROBER	通常情况下，重新生成配置文件时会调用外部程序 os-prober，发现本机上安装的其他操作系统，并产生合适的菜单项。若此项值为“true”则关闭此功能	false
GRUB_INIT_TUNE	在 grub 启动时同步的在扬声器上播放声音，选项的值被直接传递到 play 命令，如 GRUB_INIT_TUNE=“480 440 1”	无
GRUB_PRELOAD_MODULES	指定在 grub.cfg 加载之前加载的模块名，若为多个，则需用空格分隔	无
GRUB_DISABLE_LINUX_UUID	通常使用 UUID 识别根文件系统，使用 root=UUID=…作为内核参数。如果希望关闭此功能，请将此选项值设为“true”	false

5．/etc/grub.d/目录

在/etc/grub.d/目录中的所有具有执行权的文件将会在配置更新时按照文件名的 ASCII 码排列顺序被依次读取，并将其中的命令合并至配置文件 grub.cfg 中。/etc/grub.d/中常见的文件如表 7-2 所示。用户可以在此目录内添加自己的配置文件，如 11_myos。

表 7-2　GRUB2 的/etc/grub.d/目录中的常见文件

文　　件	作　　用
00_header	配置初始显示项目（如背景文件、默认项、超时时限等）。一般不需要处理此文件
10_linux	定义本系统启动项
20_linux_xen	定义第三方应用启动项。若不使用可以删除它，这里定义的是 linux_xen
30_os-prober	用于搜索 Linux 和其他操作系统。可修改/etc/default/grub 中的 GRUB_DISABLE_OS_PROBER 值来禁止对此文件的使用。
40_custom	用于加载自定义菜单项的模板。此文件的内容在注释掉 exec tail -n +3 $0 所在行后不加修改地导入到配置文件中
41_custom	用于加载/boot/grub2/custom.cfg 文件

6．grub 的部分命令

grub 的命令分为菜单、一般和命令行交互命令 3 类，常用的命令如表 7-3 所示。

表 7-3　grub 的部分命令

命　　令	功　　能
menuentry	定义一个启动菜单项
submenu	定义一个子菜单项，其中可以包含多个 menuentry

（续表）

命　　令	功　　能
Boot	启动操作系统
cat	显示文件内容（类似 shell 命令 cat）
chainloader	加载指定分区内的引导程序来引导系统
cmp	比较两个文件（类似 shell 命令 cmp）
configfile file	加载配置文件 file
date	显示或设置日期与时间（类似 shell 命令 date）
echo	显示字符串或变量的值（类似 shell 命令 echo）
export	定义并输出一个环境变量（类似 shell 命令 export）
halt	关闭计算机系统
help [iterm]	（对 iterm 进行）帮助
linux/linux16 file	加载位 Linux 内核文件 file
initrd/initrd16 file	加载启动映像并设置启动参数。在 Linux/Linux16 后使用
insmod module	动态加载模块 module
ls	列目录（类似 shell 命令 ls）
read	读取用户输入（类似 shell 命令 read）
reboot	重启系统
search [opts] name	在文件、label 或 UUID 中搜索指定的 name
set [var=value]	将环境变量 var 的值设置为 value。若无参数则显示所有环境变量
unset var	取消环境变量的值

7．配置 GRUB 示例

配置 GRUB 不容易，因为需要太多知识，至少要对 shell 编程有所了解，但是做些轻微的干预还是可以的。

当用户希望在菜单项列表后面添加新的项目时应该修改/etc/grub.d/40_custom 文件，或者在配置目录内创建 custom.cfg 文件。如果要调整启动或菜单项的顺序，请通过调整/etc/grub.d/内文件名的顺序来解决。

用户可以修改/etc/grub.d 目录下的脚本或在其中按顺序创建脚本，其中的/etc/grub.d/40_custom 通常用来添加某些菜单项（请不要修改这个文件内的头两行）。

1）直接修改配置文件

用户可以通过直接修改配置文件的办法来控制 GRUB 的启动行为，但当重新生成或更新配置文件时，被修改的内容将被覆盖。

（1）修改超时时限。

若将默认的启动等待时间改为 15s，则可通过编辑器直接打开配置文件/boot/grub2/grub.cfg，将其中的

```
set timeout=5
```

修改为

```
set timeout=15
```

后保存退出就可以了。

（2）修改默认启动项。

若要让 GRUB 默认启动第 3 项，则可做如下设置：

```
set default=2
```

这是因为项目是从 0 开始编号的。

2）修改/etc/default/grub 的内容

（1）修改超时时限。

通过/etc/default/grub 可修改 GRUB 的启动项参见表 7-1，若用同样的方法将默认的启动等待超时时间由 5s 改为 15s，则可以编辑/etc/default/grub，将其中的

```
GRUB_TIMEOUT=5
```

修改为

```
GRUB_TIMEOUT=15
```

保存后，重新生成 GRUB 的配置文件就可以。

（2）修改默认启动项。

若要让 GRUB 默认启动第 3 项，则可做如下设置：

```
GRUB_DEFAULT=2
```

然后，更新生成配置文件。此时的 GRUB_SAVEDEFAULT="true"不再具有意义。

（3）让 grub 记住并使用上次启动项作为默认启动项。

若要让 GRUB 在启动时记录本次启动项，并作为下次的临时“默认”启动项，则可修改/etc/default/grub，在其中设置如下 2 行：

```
GRUB_DEFAULT="saved"
GRUB_SAVEDEFAULT="true"
```

然后，更新生成配置文件。

3）通过 custom.cfg 修改 GRUB 的配置

为了便于修改配置，GRUB 为用户保留了一个可以方便使用的文件 custom.cfg，用户可以通过修改该文件来修改 GRUB 的配置。以下为 MSDOS 格式磁盘通过 custom.cfg 文件添加启动项的示例。

```
###grub custom.cfg
menuentry "FreeBSD 10 (on /dev/sda3)" {
    set root='hd0,msdos3'
    chainloader +1
}
```

其作用是增加一个标题为“FreeBSD 10 (on /dev/sda3)”的启动项，启动第 1 号硬盘第 3 个基本分区上的操作系统。

其实，这可作为一个模板，因为不论（hd0,msdos3）上安装的是什么操作系统一般都是

可以的。对于其他分区上的操作系统也可以采用同样的办法处理，只需要将标题、硬盘和分区编号修改正确就可以。因此，上述模板可以一般化为：

```
menuentry "Title for Some OS" {
    set root='hdM,msdosN'
    chainloader +1
}
```

其中，M 为硬盘编号；N 为分区编号。

文件 custom.cfg 并不事先存在，可通过编辑的方法创建，并根据不同的需求添加相关内容。

7.2.4　系统的启动及启动参数修改

1．启动参数的临时修改

临时修改启动参数的大致过程如下。

（1）首先加电启动系统，在超时自动启动之前，通过“↓”或“↑”选择启动项（如图 7-16 所示）。

```
Microsoft Windows XP Professional (on /dev/sda1)
Fedora Linux, with Linux ….fc24.x86_64(on /dev/sda2)
Fedora Linux, with Linux ….fc24.x86_64 (recovery mode)
FreeBSD 11 (on /dev/sda3)
```

图 7-16　系统启动的 GRUB 菜单

选中之后，按“e”进入编辑界面，选中“linux /vmlinuz-…”或“linux16 /vmlinuz-…”所在的行，再将光标移至该行末（见图 7-17 阴影部分），在行末输入至少一个空格。

```
setparms
load_video
    set gfxpayload=keep
    insmod gzio
    insmod part_msdos
    insmod ext2
    set root='(hd0,msdos2)'
    search --no-floppy --fs-uuid --set=root 2845e35c-1f41-46a2-9c31-71ee92e4f340
    echo   'Loading Linux … .fc16.x86_64 ...'
    linux   /vmlinuz-….fc24.x86_64 root=UUID=f7c4707d-4635-4d74-8c79-2a\
    68e271babf ro rd.md=0 rd.lvm=0 rd.dm=0    KEYTABLE=us quiet SYSFONT=latar\
    cyrheb-sun16 rhgb rd.luks=0 LANG=en_US.UTF-8 _
    echo   'Loading initial ramdisk ...'
    initrd  /initramfs-….fc24.x86_64.img
```

图 7-17　Fedora 的 Linux 所在的行

需要说明的是，选中的行可能会很长，从而被分隔显示在多个行中，在这些行中，除最后一行外，每个行末均有一个继续行符“\”，修改选中行时，一般只在行末添加内容。

（2）进入 rescue（系统维护）模式。

在（1）的基础上，再输入-s 或 systemd.unit=rescue.target，然后按“Ctrl-X”或 F10 键启动系统，进入 rescue 模式（以前版本无需 root 密码，直接出现提示符“#”）。当前版本系统显示为：

Welcome to rescue mode！After logging,type "journalctl -xb" to view system logs, "systemctl reboot" to reboot, "systemctl default" or ^D to boot into default mode.

Give root password for maintenance
(or type Control-D to continue):

（3）进入 emergency 模式。

在（1）的基础上，再输入-b 或 systemd.unit=emergency.target，然后按“Ctrl-X”或 F10 键启动系统，进入 emergency 模式。当前版本系统显示为：

```
Welcome to emergency mode! …
Give root password for maintenance
(or type Control-D to continue):
```

光标停在“:”后，等待用户输入信息。此时用户所能做的工作是，要么输入正确的超级用户密码进入系统维护模式；要么输入“^D”继续正常启动。不过，进入维护模式后，若没有做“原则性”修改，则仍可按“^D”继续启动到 default 模式。当然，也可以在做了原则性修改后重启系统。

2．启动参数的永久修改

按照 Linux 的传统，对以太网卡的命名采用的是 eth#办法，如 eth0、eth1 等。引入 systemd 之后，网卡的命名变了，如 eno16777736、enp7s0、ens33 等，有点让人捉摸不透。这里给出一个不常用的例子，仍将网卡名变为原来的 eth#，方法如下。

（1）编辑/etc/default/grub 文件。

找到

```
GRUB_CMDLINE_LINUX="… "
```

所在行，在行末的右双引号前添加一个空格后，再添加 net.ifnames=0 biosdevname=0 使其变为

```
GRUB_CMDLINE_LINUX="…  net.ifnames=0 biosdevname=0"
```

后保存。

（2）重新生成配置文件 grub.cfg。

进入配置文件所在目录，执行命令：

```
# grub2-mkconfig -o grub.cfg            # 红帽系统
# grub-mkconfig  -o grub.cfg            # Ubuntu 系统
```

（3）其他工作。

还要将/etc/sysconfig/network-scripts/ifcfg-eno16777736（eno16777736 也可能为其他的）

复制到/etc/sysconfig/network-scripts/ifcfg-eth0（0 为第 1 块网卡，若有多块网卡，也可能是 1 或其他），并将其中的

```
NAME=eno16777736            #（说明：eno16777736 为原来的网卡，也可能为其他的）
```

改为

```
NAME=eth0
```

然后重启系统。

7.3　软件包管理

红帽系列的系统都以 rpm 和 yum/dnf 作为软件包管理工具，Ubuntu 的软件包管理工具为 dpkg 和 apt 相关的工具，它们都将在系统安装时被自动安装。

7.3.1　红帽系统的软件包管理

1．概述

rpm（Red Hat Package Manager）是一个开放的字符界面软件包管理系统。rpm 的发布是基于 GPL 的，它最初用于 Red Hat Linux 系统的软件包管理。Red Hat 公司鼓励其他公司了解和使用它，后来成了 Linux 系统通用的软件包管理标准。

rpm 由 rpm 社区负责维护，用户可以从官方网站 http://www.rpm.org 上得到它。

1）软件包管理系统 rpm 的功能

软件管理系统 rpm 可以将软件连同源代码打成包供最终用户使用，它可以提供软件包的快速安装等管理，使用户减少编译和安装过程的困扰。rpm 软件包可以在图形或字符界面下轻松使用，其主要功能如下。

（1）查询（query）：查询软件包信息。

（2）安装（install）：将软件包的内容安装在系统的指定位置。

（3）卸载（remove）：删除或卸载指定软件包。

（4）升级或更新（upgrade）：升级或更新指定软件包。

（5）验证（verify）：检查系统中已安装的软件包与软件包中的内容是否相符。

2）软件包的分类

Rpm 软件包主要有两大类：可执行的二进制包；源程序和用于生成二进制可执行程序的源代码包。

3）软件包的获取

rpm 软件包可以从以下地方获得：发行套件的 cd/dvd 上；Linux 厂家的主站点或资源库；硬件发行商的站点；http://rpm.pbone.net。

4）rpm 包的文件名格式

rpm 包的文件名格式为：

```
pkgname-version.type.rpm
```

以 bash 包 bash-4.3.42-7.fc24.i686 为例说明如下。

（1）pkgnam 软件包名，bash。

（2）version：版本号，bash-4.3.42-7.fc24。

（3）type：i686。可能类型有 alpha（alpha 平台）、i[3456]86（Intel x86 平台）、sparc（sparc 平台）、src（源代码）和 noarch（通用，不分平台）。

（4）rpm：扩展名，不可变。

2．rpm 命令介绍

1）功能与用法

rpm 命令的参数较多，常用方法如下。

（1）rpm {-q | --query} [options]　　　　　　#查询

（2）rpm {-i | --install} [options] packages　　#安装

（3）rpm {-e | --erase} [options] packages　　#删除

（4）rpm {-U | --upgrade} [options] packages　#升级安装

（5）rpm {-F | --fresh} [options] packages　　#重新安装

（6）rpm {-V | --verify} [options]　　　　　　#校验

以上参数中 i、F、U 是功能参数，分别用于安装、重新安装、升级，e、q、V 分别用于软件包删除、查询和验证，v、h 是修饰参数，显示安装过程信息和在安装过程中显示一个进度条。对于每种功能还有一些相关参数。

2）rpm 应用举例

（1）查询。

① rpm 用于查询的常用形式有：

```
rpm -qa                      #查询所有已安装的包
rpm -q pkgname               #查询指定包
rpm -qi pkgname              #查询已安装包的详细信息
rpm -ql pkgname              #查询已安装包中的内容
rpm -qf filename             #查询指定文件的归属包
rpm -q -provides pkgname     #查询软件包功能信息
rpm -qR pkgname              #查询包的最低依赖要求
rpm -qp rpmfile              #查询 rpm 包文件中的内容
```

② 查询示例。

```
#rpm -q dhcp                 #查询是否安装了软件包 dhcp
#rpm -qa | grep sysstat      #查询是否安装了软件包 sysstat
#rpm -qi dhcp                #查询软件包 dhcp 的详细说明信息
#rpm -ql dhcp                #查询 dhcp 软件包的内容
#rpm -qf `which bash`        #查询 bash 命令的归属包
```

（2）安装。

① rpm 用于安装软件包的常用形式为：

```
rpm -ivh package             #安装软件包 package
```

② 安装示例。

```
#rpm -ivh sysstat*              #安装软件包 sysstat，需先下载到本地
#rpm -qa | grep sysstat         #查询安装结果
```

在安装过程中，我们假定 sysstat*.rpm 在当前目录内，如果不在当前目录，则可以使用带有路径的文件名，如/tmp/sysstat*。这里使用了通配符*，从而可以将所符合条件的软件包都安装上，并且在安装过程中还会按照含有*的包名来检查它们之间的依赖关系。在具体工作中，一个完整的软件包可能会依赖很多其他包，采用这种方法安装则会减少不必要的麻烦。能方便解决依赖关系，并安装依赖关系的软件包是 yum（见后述）。

（3）升级。

对指定软件包的升级可以使用以下形式的 rpm 命令：

```
rpm -Uvh packages
```

（4）删除。

对指定软件包的删除可以使用以下形式的 rpm 命令：

```
rpm -e packages
```

（5）验证。

① 软件包校验常用形式。

系统中安装的软件包可能会因为各种误操作，如修改权限、删除等造成软件包内容的损坏或不一致，因此有必要进行校验。若校验过程中出现错误输出，则说明被校验的软件包出了问题，否则软件是正确的。校验的结果也可通过命令的返回值来确定，返回值为 0 说明正常，否则是有问题的。校验软件包常用 rpm 命令形式如下：

```
rpm -Va              验证所有已经安装的包
rpm -V pkgname       验证指定包
rpm -Vf filename     验证包含指定文件的包
rpm -Vp rpmfile      验证 rpm 包文件
```

② 软件包校验实例。

```
#rpm -Va                    #校验系统所有软件的完整性
SM5....T c /var/lib/nfs/state
… … … … … …
#rpm -V sendmail            #校验 sendmail 软件包
```

③ 校验状态。

在对软件包进行校验时，若有输出则说明系统中的软件包文件与安装前文件相比发生了变化或出现了错误。在 rpm -V 的输出中有两类，一类是错误信息，如 missing somefile。另一类是校验状态信息，输出分为两列，第一列说明文件变化的原因或当前状态（见表 7-4），第二列为文件名。

表 7-4　rpm 软件包校验状态标志

状 态 标 志	意　　义	状 态 标 志	意　　义
S	大小不同	L	符号链接目标不同

（续表）

状态标志	意　义	状态标志	意　义
M	权限或类型不同	U	文件主不同
5	MD5 校验和不同	G	文件组不同
D	设备文件的主、次设备号不同	T	文件修改时间不同

3．yum 和 dnf

yum 是 Yellow Dog Linux 的一个套件，是基于 rpm 的交互式软件包管理工具，能进行系统自动升级、安装，也支持软件包的删除和查询等功能。红帽系列 Linux 的最小安装就支持 yum。使用 yum 使安装 rpm 包和升级系统变得非常简单，因此可用于升级 Linux 的各个版本。yum 正在被 dnf 所替代，不过 dnf 兼容 yum，先把 dnf 和 yum 看作同一个命令，因为一些版本还没有跟上，还在使用 yum。下面对 yum 做一些简单介绍。

1）用法

```
yum [options] [command] [package ...]
```

2）常用选项

-y：默认 yes；-q：安静工作，抑制不必要的输出；-v：与-q 相反，在工作中给出大量调试信息；

-x：反向选择，排除某些包。

3）常用命令（command）

install package(s)：安装软件包。

update [package(s)]：升级软件包，若不指定包名，则升级所有包。

check-update：检查可用更新。

upgrade [package(s)]：带有选项--obsoletes 的 update 命令。

remove/erase package(s)：删除软件包。

list [...]：列出可用包的各种信息。

info [...]：列出可用包的描述信息。

provides | whatprovides feature1 [feature2] [...]：列出特性信息。

clean [packages | headers | metadata | dbcache | all]：清除缓存信息。

localinstall rpmfile1 [rpmfile2] [...]：本地 rpm 文件安装。

localupdate rpmfile1 [rpmfile2] [...]：从本地 rpm 文件升级。

deplist package1 [package2] [...]：列出包间依赖关系。

repolist [all|enabled|disabled]：列出可用资源库。

help [command]：帮助。

4）应用示例

```
#yum list                       #列出资源库中所有软件包
#yum check-update               #检查可更新的软件包
#yum update                     #更新所有的软件包
#yum update kernel              #更新内核包 kernel
#yum install xinetd             #安装 xinetd 包
```

```
#yum remove licq                    #删除 licq 及与该包有依赖性的包
##列出资源库中特定的可以安装或更新及已经安装的 rpm 包
#yum list mozilla* telnet
##列出所有可以更新/已经安装/额外（已安装的但不在资源库中）的 rpm 包
#yum list updates/installed/extras
##列出资源库中所有可以更新的 rpm 包信息/已经安装/额外的 rpm 包信息
#yum info updates/installed/extras
```

4．yumdownloader

yumdownloader 用于从软件源代码包下载软件包，其用法为：

```
yumdownloader [options] package1 [package2...]
```

其部分可用选项如下。

--destdir DIR：设置下载目录，默认为当前目录；

--urls：显示下载位置；

--resolve：解析依赖关系，并同时下载依赖软件包；

--source：下载源代码包。

使用示例如下。

```
#yumdownloader dhcp              #下载 dhcp 包
#yumdownloader --source  dhcp    #下载 dhcp 源代码包
```

5．yum-builddep

yum-builddep 用于安装某个软件包所丢失的依赖包，没有选项。使用示例如下：

```
#yum-builddep kernel             #安装系统中还没有安装的 kernel 依赖包
```

7.3.2　ubuntu 的软件包管理

1．概述

ubuntu 采用了 Debian 的软件包管理机制。由于软件包具有易用性、灵活性和扩展性特点，再加上 Internet 的支持，使用户随时都能拥有最新的 ubuntu 系统。

通常 Debian 包文件的后缀为.deb，因此称为“deb 软件包”。ubuntu 也有两种类型的软件包：二进制软件包（.deb）和源码包（.dsc）。deb 软件包名的格式为：

```
filename_version_reversion_architechture.deb
```

其中，filename 为软件包名；version：为版本号；reversion 为修订版本号；architechture 为体系结构或类型；.deb 为扩展名。

2．管理工具

ubuntu 用于软件包管理的工具是 dpkg 和 apt 两个软件包，其中包含的常用工具有 dpkg、

apt-get 和 apt-cache 等，这里只从应用的角度出发给出它们的简单介绍。

1）dpkg

dpkg 用于本地软件包的管理，其用法为：

```
dpkg [option...] action
```

常用功能如下。

-i, --install package-file ...：安装

--unpack package-file ...：解包

-V, --verify [package-name ...]：校验

-l, --list package-name-pattern ...：列出已经安装的包，类似 rpm -qa

-L, --listfiles package-name ...：列出已经安装软件包的内容，类似 rpm -ql

-S, --search filename-search-pattern ...：显示文件归属的软件包。类似 rpm -qf

-s, --status package-name ...：报告软件包的状态信息

-r, --remove package ...：删除软件包

-P, --purge package ...：清理已经安装或被删除的软件包（包括配置文件）

示例如下。

```
dpkg -l                                        #列出所有已经安装的软件包
dpkg -l | awk '{ print $2,$3,$4 }' | grep bash #列出已经安装与 bash 相关的软件包
dpkg -L bash                                   #列出 bash 包的内容
dpkg -S `which bash`                           #查询 bash 命令的归属包
```

2）apt-cache

apt-cache 命令用于软件包缓存的管理与操作。其用法为：

```
apt-cache [-options] [action]
```

常用选项如下。

--names-only, -n：只显示名字，而非全部信息

--all-names：显示所有名字，包括虚拟包、丢失依赖关系等

--recurse：构造正向和反向依赖关系，显示所有包相关信息

--installed：仅限于已经安装的包

-h, --help：帮助

常用功能如下。

showpkg pkg ...：显示软件包信息

stats：显示软件包缓存信息

show pkg ...：显示软件包记录信息

search regex ...：查询软件包

depends pkg ...：显示软件包依赖关系信息

pkgnames [prefix]：显示以[prefix]开头的软件包名

使用示例如下。

```
#apt-cache stats
#apt-cache showpkg bash tar vsftpd
```

```
#apt-cache show bash tar vsftpd
#apt-cache search bash tar vsftpd
#apt-cache depends ftpd
```

3）apt-get

apt-get 命令的用法为：

```
apt-get [options] [action]
```

常用选项如下。

-d, --download-only：只下载

-f, --fix-broken：试图修复已破坏的依赖关系

-m, --ignore-missing：忽略丢失包

-q, --quiet：安静工作

-s, --simulate, --just-print, --dry-run, --recon, --no-act：仅模拟而不真正工作

-y, --yes, --assume-yes：默认为 yes

--assume-no：默认为 no

-u, --show-upgraded：显示被升级的包信息

--no-upgrade：安装时不升级

--only-upgrade：只升级

--show-progress：显示进度

-h, --help：帮助

--reinstall：重新安装最新版本

--purge：删除包时使用 purge，remove --purge 等同 purge

常用功能如下。

install：安装

update：更新包索引

upgrade：更新软件包

dist-upgrade：将系统升级到新版本

remove：删除

purge：删除并清理

source：获取源代码包

check：检查，更新缓存，检查可调整关系

download：下载包

autoclean：将已经删除了的软件包的.deb 安装文件从硬盘中清除

clean：删除包缓存中的所有包

autoremove：删除为了满足其他软件包的依赖而安装的、但现在不再需要的软件包

使用示例如下。

apt-get update：更新包索引。在修改源后需运行该命令，或运行该命令以确保包索引是最新的

apt-get upgrade：更新所有软件包

apt-get dist-upgrade：将系统升级到新版本

apt-get install vsftpd：安装 vsftpd 服务器

apt-get source vsftpd：下载 vsftpd 源代码。需要在 sources.list 中至少指定一个 URIs

4）apt

ubuntu 还提供有一个整合了 apt-get 和 apt-cache 的综合命令 apt，其用法为：

apt [-h] {list | search | show | update |install pkg | remove pkg ... | upgrade | full-upgrade | edit-sources | {-v | --version} | {-h | --help}}

子功能 list、show、install 和 remove 等功能分别对应 apt-cache 和 apt-get 的相关功能。

有了 apt 之后，ubuntu 的软件包管理方便多了，如查询和安装软件包 ntp*，可分别使用以下命令：

```
#apt list ntp*                  #查询 ntp*软件包
#apt install ntp*               #安装 ntp*软件包
```

7.3.3　其他格式的软件包管理

在 Linux 系统中除了提供对常见的 rpm、deb 等软件包支持及管理外，还沿袭了传统 UNIX 系统的管理办法，仍然支持 tar 和 cpio 包的管理。这两种形式的包，其文件名有时是标准的，如*.tar.gz、*.tgz、*.tzj 或*.tar 包等容易识别，但也有不易识别的，不过总能通过 file 命令识别它们是 tar、cpio 或其他类型的包，当类型确定之后，就可使用相应命令进行处理。

本部分以 flash-player 和 RealPlayer 在 Linux 系统中的应用为例说明其他形式包的安装，用户可以从相关网站（如 https://get.adobe.com/flashplayer 和 http://www.real.com/linux）下载安装包。

1．tar 格式包管理

tar 包有两种格式：压缩和非压缩格式。tar 包不能直接安装，需要时，可使用 tar 命令将包解压、展开，然后再根据包的内容来进行安装。当包被展开之后，一般来说，包内都包含说明或 README 文件可供参考。

Adobe 为 64 位系统提供的 tar 格式 flash 播放插件为 flash_player_npapi_linux.x86_64.tar.gz，可使用命令

```
#tar tvfz flash_player_npapi_linux.x86_64.tar.gz        #查询包内容
#tar xvfz flash_player_npapi_linux.x86_64.tar.gz        #提取包内容
```

展开之。在当前目录下有内容如下：

```
drwxrwxr-x. 2 root root     4096 Mar   1 08:05 LGPL
-rw-rw-r--. 1 root root 18736416 Mar   1 08:05 libpepflashplayer.so
-rw-rw-r--. 1 root root  2792147 Mar   1 08:05 license.pdf
-rw-rw-r--. 1 root root     2222 Mar   1 08:05 manifest.json
-rw-rw-r--. 1 root root     2705 Mar   1 08:05 readme.txt
```

阅读 readme.txt 得到安装指导信息：Unpack the plugin tar.gz and copy all the files to /usr/lib/adobe-flashplugin/。执行命令

```
#mkdir -p /usr/lib/adobe-flashplugin/
#cp  * /usr/lib/adobe-flashplugin/      #或 cp -r * /usr/lib/adobe-flashplugin/
```

完成安装。

2．其他 rpm 包

对于其他不在发行套件内的 rpm 包，或其他第三方 rpm 包，可通过存储介质或网络得到软件包文件，然后通过软件包管理的办法进行安装。但是需要注意的是，当一个包依赖另外的包时，这些包文件也应该存放在同一目录内，否则安装可能失败。比如，Adobe 公司提供的 flash_player rpm 包为 flash-player-npapi-25.0.0.127-release.i386/x86_64.rpm，可在下载后使用命令

```
rpm -ivh flash-player-npapi-25.0.0.127-release.i386.rpm        #32 位系统
rpm -ivh flash-player-npapi-25.0.0.127-release.x86_64.rpm      #64 位系统
```

安装它，这要比使用.tar.gz 格式包方便一些。

3．第三方 rpm 包源

第三方 rpm 包也可通过安装第 3 方软件包源的办法来实现。比如，Adobe 公司提供有自己 rpm 包源软件包，64 位系统为 adobe-release-x86_64-1.0-1.noarch.rpm，32 位系统为 dobe-release-i386-1.0-1.noarch.rpm，若想通过 yum 和网络在线使用 Adobe 公司软件包，则要先下载并安装之。

安装之后，就可以使用命令

```
yum list adob* flash*
```

查询相关软件包，可以看到的包有 flash-player-ppapi、flash-plugin、flashrom 等，若要安装它们，使用命令

```
yum install flash-player-ppapi flash-plugin
```

就可以了，不需要再使用*.tar.gz 的软件包。

4．其他格式包

其他格式软件包还有多种，要根据经验来处理。这里以 realplayer 播放插件为例说明之。

Linux 可支持的包为 RealPlayer11GOLD.zip，解压后得到

```
RealPlayer11GOLD.bin
```

使用命令

```
#file RealPlayer11GOLD.bin
```

确定文件的类型，得到以下输出：

```
RealPlayer11GOLD.bin: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.4.1, not stripped
```

说明这是一个可执行文件，为它增加执行权，再执行之：

```
#chmod +x RealPlayer11GOLD.bin
#./RealPlayer11GOLD.bin
```

然后按提示要求进行操作就可以。不过，由于是 32 位的，只能在 32 位环境下安装，且安装前还要安装 pangox-compat 包。

5．cpio 包

cpio 包不能直接安装，需要使用 cpio 命令将其解压、展开，然后再根据包的内容进行安装。展开的方法为：

```
#cpio -i -t < cpio_pkg            #查看包内容
#cpio -iv < cpio_pkg              #提取包内容
```

7.3.4　图形界面

Linux 的桌面或工作站系统都有图形界面，在图形界面下都提供软件包管理系统，用于本系统的软件包管理。可在 CLI 界面下通过命令

```
#gnome-software                   #软件包管理工具
```

来启动。在它的控制下，可以完成软件包的安装、升级与删除等管理工作。

7.4　系统升级

操作系统的自我完善是必需的，一个有生命力的操作系统都提供有某种方式的自我完善机制，现在的 Linux 都有这个功能。用户可在图形或字符界面下对整个系统或单个软件包进行升级，也可在系统有升级提醒时启动升级。

1．图形界面

在图形界面下，可以从 CLI 界面执行以下命令启动软件包管理和升级工具：

```
#gnome-software          #软件包管理工具：需要 gnome-software 软件包支持
#gpk-update-viewer       #软件包升级管理器：需要 gnome-packagekit-updater 软件包支持
```

2．红帽系统的升级

红帽系统的网络软件包管理工具为 yum 或（将来的）dnf，用于升级的用法为：

```
#y um check-update       #检查可更新的软件包
#yum update              #更新所有的软件包
```

3．ubuntu 系统的升级

ubuntu 系统的网络软件包管理工具为 apt-get，用于升级的用法为：

```
#apt-get update          #更新包索引
#apt-get upgrade         #更新所有软件包
#apt-get dist-upgrade    #将系统升级到新版本
```

7.5　日志管理

日志的主要用途是记录系统操作的过程，是一个流水账，它记录了系统中每周、每天、每时、每刻发生的各种各样的事情。通过它，用户可以检查错误发生的原因，或者受到攻击时攻击者留下的痕迹，以便对系统行为进行跟踪、监视和统计分析。也就是说，日志主要的作用是用于审计和监测。为了保证系统的正常运行和出现故障时分析有据，准确地记录日志是非常必要的。

现在书中所涉及的系统使用的都是 rsyslog（reliable and extended syslog）日志系统，而以前使用的是 syslog 日志系统。

7.5.1　日志系统

1．日志系统概述

Linux 系统的日志系统是一个综合的日志管理系统，通过设置能将不同信息输出到不同的日志文件，从而实现信息分类存放与管理。

Linux 系统的日志由两个主要的子系统组成。

（1）登录日志：login 等程序更新 wtmp、btmp 和 lastlog 文件，使系统管理员能够跟踪谁在何时、何地登录和使用系统。

（2）系统日志：由 rsyslogd 实施。各种系统守候进程、用户程序和内核通过 rsyslog 向/var/log/messages 等日志文件报告值得注意的事件。（注：之前版本为 syslogd）

另外，有许多像 HTTP 和 FTP 这样提供网络服务的服务器或应用程序也都记录有详细的工作日志，而 sar、iostat 等则记录系统的 CPU、I/O 等系统活动情况。

rsyslog 系统日志的守候进程为 rsyslogd，服务名为 rsyslog.service。

2．日志配置文件

/etc/rsyslog.conf 为 rsyslogd 的主配置文件，它规定系统中需要监视的事件和记录相应的日志位置，描述未尽部分存放在/etc/rsyslog.d 的子配置文件中。为了对日志统一管理，系统的各种日志文件存放在/var/log 目录内。管理员可以通过编辑配置文件 rsyslog 的办法来影响系统日志的行为。

日志配置文件每一有效行的格式如下：

```
选择　动作
```

可以进一步解释为：

```
功能.级别 动作
```

功能、级别和动作的意义参见表 7-5。在“功能”和“级别”部分均可使用配置符“*”表示所有允许的值。在配置文件中的一个有效行的选择部分可以是用分号连接的多个“功能.级别”对的组合。可通过阅读 rsyslog.conf 手册页做深入了解。

表 7-5　/etc/rsyslog.conf 的功能、级别和动作的意义

字 段	子 字 段	意 义
功能	authpriv	认证活动、口令等私有信息不记录
	cron	crond 与 atd 等作业调试信息
	daemon	系统 xinetd 守候进程信息
	ftp	ftp 守候进程信息
	kern	内核信息
	lpr	打印服务信息
	mail	邮件服务信息
	mark	系统内部信息
	news	新闻有关信息
	syslog	syslog 信息
	user	由用户程序生成的信息
	uucp	UUCP 系统生成的信息
	local0～local7	为本地应用程序保留
	*	除 mark 外的所有信息
级别	alert	需要立即引起注意并采取措施的情况
	crit	紧急情况
	debug	在运行于 DEBUG 模式产生的信息
	emerg	紧急情况，系统不可用
	err	错误情况（除 emerg，alert 和 crit 以外）
	info	信息型消息
	none	用于禁止某类消息
	notice	正常的，但是重要的情况（不及 err 和 warning 等）
	warning	警告信息
	*	除了 none 以外的所有级别
动作	file	指定一个含有绝对路径的文件名，可以是命名管道或终端设备，如控制台等
	主机名	指定一个运行 syslogd 并接收 syslog 信息的主机。可以用此特性将 syslog 信息 log 到某个主机上
	用户名	将信息发往指定用户，当用户不止一个时用逗号分隔，*代表所有用户

3．配置示例

（1）除内核信息外，将所有 critical 信息写到日志文件/var/log/critical 上。

```
*.=crit;kern.none                                    /var/log/critical
```

（2）除了 mail、news、authpriv 和 cron 外，将 info 或更高级别的信息送到/var/log/messages 上。

*.info;mail.none;news.none;authpriv.none;cron.none/var/log/messages

（3）将 mail 系统信息记录到/var/log/maillog。

```
mail.*                                    /var/log/maillog
```

（4）将所有 emerg 信息以 wall 的形式发送给系统内的所有登录用户。

```
*.=emerg                                  *
```

（5）将所有日志信息送往主机 loghost。

```
*.*                                       @loghost
```

（6）将 news 服务的 crit、err 和 notice 分别 log 到/var/log/news/news.*。

```
news.=crit                                /var/log/news/news.crit
news.=err                                 /var/log/news/news.err
news.notice                               /var/log/news/news.notice
```

7.5.2　常见日志文件及阅读

1．常见日志文件

日志文件的名字和位置由配置文件/etc/rsyslog.conf 决定，通常情况下，存放在/var/log 目录内。常见的日志文件如表 7-6 所示。

表 7-6　常见的用日志文件

日 志 类 型	文　件　名	说　　明
boot	boot.log	系统启动日志
cron	cron	守候进程 crond 日志
dmesg	dmesg	内核启动日志
lastlog	lastlog	系统 info 信息
mail	maillog	senadmail 日志
*.info	messages	系统 info 信息，系统日志
authpriv	secure	系统认证信息，安全日志
login	wtmp	用户登录和退出信息

2．日志阅读

有两种形式的日志文件：纯文本文件和非文本文件。对于纯文本型的日志文件可使用文本编辑与查看软件进行阅读，可用的工具有 vi、head、tail、less 和 grep 等；而对于非文本日志文件则需要使用专用软件来阅读。

1）文本型日志文件阅读

除内核启动日志 dmesg 等少数日志文件外，一般文本型日志文件的格式为每行一条信息，每个信息由 4 个域组成。

（1）时间戳：信息产生的日期和时间。
（2）主机名：产生信息的计算机。
（3）日志子系统：生成日志信息的日志子系统。
（4）消息内容：本条消息的具体内容。
以下为文件 message 的两行内容：

```
Feb 22 19:18:52 hncj named[1656]:listening on IPv4 interface eth0:0, 192.168.1.5#53
Feb 22 23:03:41 hncj login(pam_unix)[3305]: check pass; user unknown
```

第一行说明：信息产生于 Feb 22 19:18:52，主机为 hncj，信息源于 named，内容为 listening on IPv4 interface eth0:0, 192.168.181.5#53。

第二行说明：信息产生于 Feb 22 23:03:41，主机为 hncj，信息源于 login，内容为 check pass;user unknown。

文本型日志还可在图形界面上阅读，方法是在 CLI 界面下执行命令：

```
#gnome-system-log
```

也可以直接进入/var/log 目录，找到相关日志文件，然后使用文件编辑与处理工具阅读和分析它们。有的系统还提供有 loganalyzer，用于日志文件的分析。

2）非文本日志文件阅读

非文本日志文件有/etc/log/wtmp、/var/log/btmp、/var/log/lastlog 和/var/run/utmp 等，它们记录着用户登录的相关信息，可以使用 lastlog、last、lastb、who 和 w 来阅读。

（1）lastlog。

lastlog 根据文件/var/log/lastlog 内容显示系统中所有或指定用户的最近登录时间。其用法为：

```
lastlog [-b DAYS] [-t DAYS] [-u user] …
```

参数-b 用于指定几天前的；-t 用于指定几天后的；-u 用于指定用户。

以下为 lastlog 命令的输出。

```
#lastlog                                 #显示所有用户的最后登录时间
  Username     Port     From             Latest
  root         tty2                      Mon Mar 20 14:19:19 +0800 2017
  bin                                    **Never logged in**
  gdm          tty1                      Mon Mar 20 14:17:34 +0800 2017
  rpcuser                                **Never logged in**
  test         tty2                      Sat Mar 18 10:33:51 +0800 2017
  hncj         tty4                      Wed Mar 15 20:55:31 +0800 2017
  mytest       tty3                      Sat Mar 18 10:41:34 +0800 2017
#lastlog -u root                         #显示指定用户的最后登录时间
#lastlog -t 3 -u 500-510                 #显示指定用户群的最后登录时间
```

（2）last。

/var/log/wtmp 文件记录有系统所有用户的登录信息，last 用来显示自文件/var/log/wtmp 产生以来的用户登录信息。last 的用法为：

```
last [-R|-a] [-num] [-n num] [diox] [-f FILE] [-s TIME] [-t TIME] [name] [tty]
```

-R 和-a 分别用于禁止或允许输出主机名；-num 或-n num 用于指定显示的行数；-x 用于指定只显示系统关闭和运行级切换的内容；-f 用于指定文件 FILE 而取代默认文件；-s 和-t 用于指定时间开始和结束时间（时间格式为：YYYYMMDDhhmmss、YYYY-MM-DD hh:mm:ss、YYYY-MM-DD hh:mm、YYYY-MM-DD、hh:mm:ss 和 hh:mm 等）；name 用于指定用户；tty 用于指定终端设备，且可缩写，如 tty1 可写为 1。若时间、用户名或终端名指定，则只显示与之匹配的内容。

last 使用示例如下。

```
#last -10                                  #显示最后 10 条登录信息
#last -s 2017-03-11                        #显示从 2017 年 3 月 1 日开始的登录信息
#last -s 2017-03-11 -t 20170320000000 root
```

（3）lastb。

lastb 用于显示试图登录系统但未成功者的信息，其用法与 last 相同，所不同的是 lastb 使用的文件是/var/log/btmp。以下为使用 lastb 查询登录不成功的用户信息。

```
#lastb -n 10                               #显示最后 10 条登录不成功信息
#lastb -s 2017-03-11 -t 2017-03-20 root    #显示在这两个日期间 root 登录不成功信息
```

文件/var/log/wtmp 和/var/log/btmp 两者均可能不存在，若不存在，则 last 或 lastb 不能正常使用。需要时，可以使用命令

```
#touch /var/log/wtmp /var/log/btmp
```

创建它们。

3）内核启动日志

系统开机时会将内核启动信息存储在 ring buffer 中，dmesg 用来检查和控制 ring buffer。dmesg 的默认动作就是显示所有来自 ring buffer 的信息。

通过查看内核启动日志，可以了解系统的启动情况及与系统连接的设备情况，dmesg 应用示例如下：

```
#dmesg                 #显示所有信息
#dmesg|grep -i pci     #查看含有 pci 字符串的信息（了解 PCI 总线情况）
#dmesg|grep -i usb     #查看含有 usb 字符串的信息（了解 USB 总线情况）
#dmesg|grep disk       #查看含有 disk 字符串的信息（了解硬盘情况）
#dmesg|grep cd         #查看含有 cd 字符串的信息（了解光驱情况）
#dmesg|grep net        #查看含有 net 字符串的信息（了解网卡情况）
```

7.5.3　日志滚动

随着系统的运行，日志会越记越多，这样就带来至少两方面问题：一是日志的增多会占用大量磁盘空间，二是大文件的存在对阅读、处理等操作带来不利。还有一个问题就是很久很久以前的东西也没有必要保留在现行日志文件里供考古使用。

对日志文件进行处理最自然的办法是去掉久远的内容，使用各类日志文件限制在可以方便阅读的范围内，时间保持在最有效的范围内，于是就出现了滚动记载日志的想法。

命令 logrotate 负责日志的滚动，它作为 crond 的一个日常性工作来对日志文件进行自动滚动、压缩、删除或将日志作为电子邮件发送。

关于日志滚动，系统安装时已经有默认设置，可以满足使用要求，用户一般不需改动。

7.6 系统管理

系统管理是系统管理员的事情。到目前为止，我们学习的很多内容都和系统管理联系着，如用户管理、进程管理、文件系统管理等，也包括设备、网络管理和网络安全等。系统管理的主要任务是利用各种管理功能让系统运行得更好、更高效。

7.6.1 系统管理的任务

每个 Linux 系统应至少有一个人负责系统的维护与日常管理，这个人就称为系统管理员。系统管理员的职责是规划、安装和维护系统，确保系统的高效、平稳运行。系统管理员主要负责：

（1）系统的开启与关闭及对启动过程进行维护。

（2）数据备份并保存好备份数据。

（3）处理器与计算机有限资源的利用问题。

（4）系统升级、扩充等维护和修补。

（5）设备管理与维护。

（6）网络管理与维护。

（7）系统安全与网络访问控制。

（8）对用户提供一般性服务及培训等。

系统管理的经常性工作如表 7-7 所示，除此之外，还有月任务、年任务等。

表 7-7 系统管理的经常性工作

任务周期	任务
日常事务	数据备份
	检查使用级别
	检查失控的进程
	检查磁盘空间
	检查网络连接
	检查打印机状态（若配置）
	检查审计输出（若配置）
	检查远程登录者（telnet、ftp 等）
周任务	校验系统软件的完整性
	检查打印机假脱机状态报告
	日志维护，日志文件阅读、清除、裁剪、截断等
	产生系统活动报告
	生成磁盘使用情况报告
	删除临时文件，包括 lost+found、/tmp 和/usr/tmp 目录内的文件

（续表）

任 务 周 期	任　　务
不定期任务	密码维护
	系统升级与软件包维护
	检查系统的软硬件配置
	重新分配文件系统空间
	寻找 suid 或 sgid 文件，检查文件主、组和文件大小
	查找大文件，并确定其用途
	查找“孤儿”文件（所有者已删除，且不再使用）

7.6.2　系统管理的工具与命令

1．显示系统信息（uname）

1）a 功能及用法

uanme 的功能是显示系统信息，其用法为：

```
uname [ options ]
```

2）参数说明

uname 的参数如表 7-8 所示。

表 7-8　uname 的参数

参　　数	意　　义	参　　数	意　　义
-a,--all	所有系统信息	-m, --machine	机型信息
-s, --kernel-name	内核名称	-p, --processor	处理器信息
-n, --nodename	网络节点名	-i, --hardware-platform	硬件平台
-r, --kernel-release	内核 release	-o, --operating-system	操作系统类型
-v, --kernel-version	内核版本号	--help	显示帮助信息

2．显示系统配置信息（getconf）

getconf 用于获取当前系统的配置信息，其用法为：

```
getconf -a
getconf [-v spec] system_var
getconf [-v spec] path_var pathname
```

选项-a 用于显示所有系统配置变量；-v spec 指定版本（这里不考虑它，使用默认）；system_var 用于显示系统变量；pathname 为指定的目录；path_var 为与目录 pathname 相关的配置变量。与系统相关的配置如表 7-9 所示。

表 7-9　与系统相关的配置

	配置变量	意　　义
系统级	ARG_MAX	使用 exec()执行命令时命令行参数最大长度
	CHILD_MAX	每个用户的最大并发进程数
	HOST_NAME_MAX	hostname 的最大长度
	LOGIN_NAME_MAX	登录用户名的最大长度
	NGROUPS_MAX	最大附加组成员数
	OPEN_MAX	进程可同时打开数
	PAGESIZE	页长度
	STREAM_MAX	进程的最大流数
	TTY_NAME_MAX	终端名最大长度
	TZNAME_MAX	时区名最大长度
	LINE_MAX	行的最大长度
	POSIX2_C_DEV	是否支持 POSIX.2 C 开发
	POSIX2_FORT_DEV	是否支持 POSIX.2 FORTRAN
	POSIX2_FORT_RUN	是否支持 POSIX.2 FORTRAN run-time
	POSIX2_SW_DEV	是否支持 POSIX.2 软件开发
目录级	LINK_MAX	文件最大链接数
	MAX_CANON	格式输入行最大长度
	MAX_INPUT	输入行最大长度
	NAME_MAX	文件名的最大长度
	PATH_MAX	目录最大长度
	PIPE_BUF	管道（pipe）缓冲区长度
	CHOWN_RESTRICTED	是否可以使用 chown（0-可，其他不可）
	NO_TRUNC	是否可以访问文件长度超过 NAME_MAX 的文件（0-可，其他不可）
	VDISABLE	特殊字符的处理是否可以启用（0-可，其他不可）

getconf 的使用示例如下。

```
#getconf -a                              #获得所配置信息
#getconf -a | grep -i LINE_MAX           #获得所有 LINE_MAX
#getconf PAGESIZE                        #获取系统的页面大小
#getconf LINK_MAX /                      #得到/的 LINK_MAX
#getconf LINK_MAX /dev                   #得到/dev 的 LINK_MAX
```

3．显示系统活动信息（uptime）

uptime 用于显示系统活动信息，其用法为：

```
uptime [ -V ]
```

uptime 显示的信息包括：

（1）当前时间；
（2）系统启动后所经历的时间；
（3）系统中的当前用户数；
（4）系统负载在最近 1 分钟、5 分钟和 15 分钟内的负载平均值。
以下是 uptime 命令的输出：

```
11:28:39    up    3:09,    5 users,    load average: 0.34, 0.22, 0.24
```

说明系统的当前时间为 11:28:39，系统启动后已经运行了 3 小时 9 分钟，现在系统内有 5 个用户，系统在 1 分钟、5 分钟和 15 分钟内的平均负载分别为 0.34，0.22 和 0.24。

4．显示虚拟内存使用情况（vmstat）

vmstat 用来监视虚拟内存使用情况，也包括有关进程状态、内存、页面、块 I/O、陷阱和 CPU 的活动情况。其用法为：

```
vmstat [options] [delay [count]]
```

delay 设定两次统计的时间间隔，count 指定总共统计的次数；-n 让 vmstat 只显示一次标题。比如，-a 将使得 vmstat 以 inact/active 替换 buff/cache；-f 将显示自从启动后执行 fork() 的次数；-m 将使得 vmstat 显示 slabinfo 信息（参见在线手册）；-s 将显示系统内事件和内存使用情况统计表；-d 将报告磁盘使用情况统计。
在不带有选项时，vmstat 统计信息包括：
（1）进程信息：r—等待运行的进程数；b—等待 I/O 而睡眠的进程数。
（2）内存信息：swpd—交换区总量；free—空闲内存；buff—用于缓冲区的；cache—用于 cache 的内存。
（3）交换信息：si—从交换区换入的内存量；so—从内存换出的内存量。
（4）I/O 信息：bi—送往块设备的信息；bo—从块设备读入的信息。
（5）系统信息：in—每秒（含时钟中断）的中断次数；cs—每秒进程上下文切换次数。
（6）CPU 信息：us—用户空间平均执行时间；sy—系统空间平均执行时间；id—空转时间；wt：等待 I/O 的 CPU 时间。
以下是命令 vmstat 5 3 的输出。在有效输出的第一行是从启动到现在的平均值，不具有参考价值，所以阅读信息要从第二行开始。

```
#vmstat 5 3

procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0  43056  23956  37864 318152    0    0     0     0    0    0  0 100  0  0  0
 0  0  43056  24064  37864 318156    0    0     0     0  120  244  0  0 99  0  0
 0  0  43056  24064  37864 318156    0    0     0     0  234  399  1  1 98  0  0
```

5．内存使用情况查询（free）

free 命令用于系统内存使用情况的查询，包括空闲内存、已用内存、用于交换的内存，以及被内核使用的缓冲区和缓存信息，其用法为：

```
free [options]
```

常用选项有：-c#-显示次数为#；-s#-次数间隔为#秒；-h：便于阅读方式；-b/--bytes、-k/--kilo、

-m/--mega、-g/--giga 和--tera 分别为 B、K、M、G 和 T。

free 从/proc/meminfo 收集以下信息：

（1）total：总内存（内存和交换）。

（2）used：已经使用。

（3）free：空闲内存。

（4）shared：共享内存（主要是被 tmpfs 使用的内存）。

（5）buffers：被内核缓冲使用的内存。

（6）cache：被页缓存和 slabs 使用的内存。

（7）buff/cache：buffers 与 cache 的和。

（8）available：启动新进程时（而不需要交换）的可用内存。

示例如下：

```
#free                          #查询内存使用情况
#free -c 2 -s 3 -h             #连续查询两次，时间间隔 3 秒，易读方式显示
```

6．显示系统中谁在工作（w）

w 用于显示谁在系统中和他们在干什么，它的用法为：

```
w [-husfV] [user]
```

以下为不带参数的 w 命令的输出结果：

```
12:39:16   up   4:20,   5 users,   load average: 0.32, 0.21, 0.16
USER       TTY       LOGIN@     IDLE    JCPU    PCPU      WHAT
root       tty1       8:21am     0.00s   1.73s    0.14s      w
… … … …
```

在输出的第一行与 uptime 的输出相同。在以下输出中有 8 个域，它们分别是：

（1）USER：用户名；

（2）TTY：用户使用的终端；

（3）LOGIN：登录时间；

（4）IDLE：休眠时间；

（5）JCPU：与终端相关的所有进程的执行时间；

（6）PCPU：当前进程占用的 CPU 时间；

（7）WHAT：正执行的进程。

7．显示系统内进程活动信息（top）

top 的功能是实时监控系统内进程活动信息，也包括 CPU 利用率、进程状态、内存利用率等信息，为系统管理员提供实时监控系统的工具。下面是不带任何参数和选项执行 top 命令监视系统运行某时刻的快照。可按^C 退出 top。

```
top - 22:29:34 up 29 min,  2 users,  load average: 0.00, 0.00, 0.12
Tasks: 193 total,   1 running, 192 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.4 us,  0.8 sy,  0.0 ni, 96.4 id,  0.5 wa,  0.5 hi,  0.3 si,  0.0 st
KiB Mem :  1023764 total,    29204 free,   636320 used,   358240 buff/cache
KiB Swap:  2097148 total,  2055188 free,    41960 used.   328564 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1617 root      20   0  553460  70344  30316 S   4.3  6.9   0:12.47 gnome-shell
 1515 root      20   0  136876  16404  10256 S   2.3  1.6   0:08.85 Xorg
  723 root      20   0   53108   4536   4316 S   0.3  0.4   0:01.43 vmtoolsd
 1545 root      20   0   15100   3840   3152 S   0.3  0.4   0:00.33 dbus-daemon
 2389 root      20   0    8768   3856   3316 R   0.3  0.4   0:00.09 top
    1 root      20   0   45036   5928   4496 S   0.0  0.6   0:02.63 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.22 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S   0.0  0.0   0:00.76 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
   10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
```

第一行与 uptime 输出相同，第二行为系统内进程信息，第三行为与 CPU 相关的信息，第四行为内存情况，第五行为交换区使用情况，第七行及以后行为系统内按活跃程度排序的进程信息。

top 命令还有一些参数可以使用，以控制工作过程和行为，请参考 top 的 man 手册。

8．监视系统 I/O 等活动情况（sar、iostat、mpstat 和 pidstat）

Linux 系统还提供有用于系统状态监视的软件包 sysstat，用户需要时，可以下载并安装它。在 sysstat 包中包括 4 个应用程序，分别是 sar、iostat、mpstat 和 pidstat。mpstat 用于监控系统内每个 CPU 的活动情况报告；pidstat 用于监督进程信息；iostat 用于报告系统内 CPU、设备和物理分区的 I/O 活动情况，根据 iostat 报告信息可以调整系统配置参数，在物理设备间找到更好的 I/O 平衡；sar 命令是一个能综合监视系统各种活动信息的程序，可用于系统各种信息的报告。sar 系列命令用到/var/log/sa/saDD 等日志文件，其中 DD 为工作日期。命令 sar、iostat、mpstat 和 pidstat 用法可统一为：

```
cmd [ options ] [ interval [ count ]]
```

它们均需要设置一个以秒为单位的时间区间（interval）和在这个区间内的报告次数（count），若不指定，则这几个命令的默认值并不相同。由于参数较多，不再做逐一介绍。

sar 命令可对系统的多项指标进行综合监视，这些指标及控制参数如表 7-10 所示。

表 7-10　sar 命令的部分参数

参　　数	意　　义	参　　数	意　　义
-A	所有参数	-u	CPU 使用信息
-b	I/O 传输率信息	-U	指定的 CPU 信息（多 CPU 时）
-B	页交换活动信息	-v	i 节点、文件和内核表状态信息
-c	进程创建信息	-w	进程上下文切换信息
-d	报告每块设备活动信息	-W	页交换信息
-I irq	中断 irq 的信息	-x	指定进程信息
-q	系统进程队列信息和平均负载信息	-X	指定进程的子进程信息
-r	内存和交换空间信息	-y	终端设备活动信息
-R	内存信息		

由表可以看出，sar 对系统的监督是广泛的，通过对这些统计信息的观察分析可以了解一个系统的运行情况，再根据情况可以对系统的软硬件配置进行调整或优化。

sysstat 包内命令的使用示例如下。

1）mpstat 示例

```
#mpstat 5 3
Linux 4.9.12-100.fc24.i686+PAE (fc24x64.unix.gjshao)    03/21/2017      _i686_     (2 CPU)
10:53:51 PM CPU %usr   %nice   %sys   %iowait %irq   %soft   %steal   %guest %gnice %idle
10:53:56 PM   all   0.30    0.00   0.40     0.00    0.50     0.10    0.00    0.00    0.00   98.69
10:54:01 PM   all   0.00    0.00   0.00     0.00    0.00   100.00    0.00    0.00    0.00    0.00
10:54:06 PM   all   0.40    0.00   1.41    26.10    0.30     0.30    0.00    0.00    0.00   71.49
Average:        all   0.00    0.00   0.00     0.00    0.00   100.00    0.00    0.00    0.00    0.00
```

说明：%usr 为用户空间执行时间；%nice 为用户空间以优先数 nice 级的执行时间；%sys 为系统空间执行时间；%iowait 为硬盘 I/O 等待时间；%irq 为硬中断时间；%soft 为软中断时间；%idle 为 CPU 闲置时间。

2）iostat 示例

```
#iostat 5 3
Linux 4.9.12-100.fc24.i686+PAE (fc24x64.unix.gjshao)    03/21/2017      _i686_     (2 CPU)
avg-cpu:   %user     %nice %system %iowait    %steal     %idle
               0.00       0.00   100.00      0.00       0.00       0.00
Device:                 tps      kB_read/s      kB_wrtn/s      kB_read      kB_wrtn
sda                    30.02         429.11          101.52      1489897       352469
dm-0                  34.55         424.75           69.98      1474781       242980
dm-1                   8.28           2.11           31.53         7332       109480
```

说明：tps 为每秒传输数；kB_read/s 为每秒读出的块数；kB_wrtn/s 为每秒写出的块数；kB_read 为读入总块数；kB_wrtn 为写出总块数。

3）sar 使用示例

```
#sar -u 2 5                              #在 2 分钟内报告 5 次 CPU 信息
#sar -r -n DEV -f /var/log/sa/sa22       #输出 22 号 sar 记录文件中的内存、交换和网络活动信息
#sar -A                                  #报告记录在当天 sar 记录文件中的所有信息
```

9．系统监视器

从 CLI 界面执行命令 gnome-system-monitor 可打开系统监视器界面，该界面共有“Processes”、“Resources”和“File Systems”3 个选项卡，分别用于对系统基于信息、进程信息、资源和文件系统的监视。

7.7　内核配置与参数在线调整

在内核安装与处理方面，不同的 Linux 系统，或相同系统不同的版本间存在着差异。为了方便起见，仅以 fedora 24 及其软件包源提供的内核包 kernel-4.9.13-101.fc24.src.rpm 为例说明配置和编译过程。

7.7.1 内核配置、编译与安装

1．下载和安装内核源代码

1）安装所需软件包

```
#yum install kernel-devel                    #安装 Kernel Headers
#yum install yum-utils rpmdevtools           #编译内核源代码包需要这两个工具
#rpmdev-setuptree                            #生成源代码安装目录（~/rpmbuild 子目录）
```

2）下载源代码包

```
#yumdownloader --source kernel               #下载源代码到~/rpmbuild/
```

说明：内核源代码包也可从其他渠道获得，如 http://rpm.pbone.net/等。

3）安装内核依赖关系文件

```
#cd ~/rpmbuild/                              #切换到源代码文件所在的目录~/rpmbuild/
#yum-builddep KERNEL_SRC
```

KERNEL_SRC 为刚刚下载的内核源代码包，如 kernel-4.9.13-101.fc24.src.rpm。具体可写为

```
#yum-builddep kernel-4.9.13-101.fc24.src.rpm        #或
#yum-builddep `ls kernel-*.src.rpm`
```

4）创建所需的组和用户（可选）

```
#groupadd mockbuild                                 #创建 mockbuild 组（可选）
#useradd mockbuild -g mockbuild                     #创建用户 mockbuild（可选）
```

5）安装源代码包

```
#rpm -ivh kernel-4.9.13-101.fc24.src.rpm            #安装内核源代码包
```

6）若需要，安装 numactl-devel 和 pesign 软件包

```
#yum install numactl-devel pesign                   #安装 numactl-devel 和 pesign 包，可合并于 1）
```

7）生成源代码

源代码包安装后还不能直接使用，还需要做如下处理：

```
#rpmbuild -bp --target=$(uname -m) ~/rpmbuild/SPECS/kernel.spec
```

成功执行后，源代码所在目录为~/rpmbuild/BUILD/kernel-4.9.fc24，在其下有两个目录：linux-4.9.13-101.fc24.x86_64 和 vanilla-4.9。前者是包含有补丁和升级包的内核源代码，后者只是原始的内核源代码。至此源代码就可以使用了。另外，与本机系统相关的补丁和更新内容存放的真正位置是~/rpmbuild/BUILD/kernel-4.9.fc24/linux-4.9.13-101.fc24.x86_64/arch/x86。

2．内核配置

内核配置是通过 make 命令实现的。make 命令依赖源代码目录内的 Makefile 文件，该文件规定了 make 的行为（参见第 12 章）。若配置内核，则首先要进入内核所在目录~/rpmbuild/BUILD/kernel-*/linux-*/。

重新配置内核的常用方法有：

```
make config          #普通文本界面
make menuconfig      #彩色菜单界面
make nconfig         #增强彩色菜单界面
make xconfig         #基于 Qt 的配置界面
make gconfig         #基于 GTK+的配置界面
make oldconfig       #基于原配置文件.config 的配置
make olddefconfig    #旧项目采用原配置，新项目采取默认值
make defconfig       #根据 arch/$ARCH/configs/${PLATFORM}_defconfig
  #或 arch/$ARCH/defconfig 生成.config 配置文件。$ARCH 为 x86，${PLATFORM}为 i386 或 x86_64
```

更多的配置命令可参考内核配置目录内的 README 文件。

内核配置过程大致如下。

（1）进入配置目录。

```
#cd ~/rpmbuild/BUILD/kernel-4.9.fc24/linux-4.9.13-101.fc24.x86_64/
```

（2）清理临时文件（可选）。

如果愿意，还可运行命令 make distclean 或 make mrproper 清除临时文件。

（3）选择配置方式。

① 参照老内核配置文件配置新内核。

建议参照老内核配置文件/boot/config-*（其中之一或其中有最新者）配置新内核，对于新手来说这样要容易一些。当然，有特殊目的，如需要增加或删除特殊配置的除外。此时可执行命令：

```
#cp /boot/config-`uname -r`* .config      #复制旧的内核配置文件到.config
#make oldconfig      #基于原配置文件.config 的配置。对原配置直接采用，只对新配置项进行提问
或   #make olddefconfig              #旧项目采用原配置，新项目采取默认值
```

② 重新配置新内核。

重新配置新内核可使用命令 make config 和 make menuconfig 等进行。

像 make config 等文本界面的交互式工具，在配置过程中需要用户回答很多问题，问题的答案有三类。

- y：将该项功能配置在内核中。
- n：将该项功能不配置在内核中。
- m：将该项功能不配置在内核中，但配置为可加载模块，需要时动态加载该模块。

由于需要配置的参数很多，且需要很多专业知识，对于非专业人员来说，用起来非常不便，因此内核配置时推荐使用菜单方式。输入命令

```
#make menuconfig
```

进入菜单配置主界面（如图 7-18 所示）。

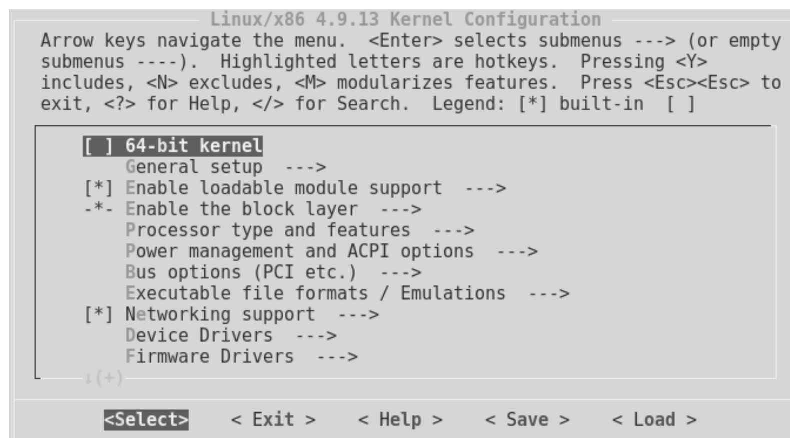


图 7-18　make menuconfig 主菜单

在如图 7-18 所示的主菜单中，可分别对不同子系统的参数进行设置。用户可以通过“↓”或“↑”和 Space 键选择不同项目而进入二级子菜单，也可通过“←”或“→”和 Space 键选择“Select”、“Exit”或“Help”。

进入二级子菜单后，用户对其中的项目可通过 Space（n，不配置）、*（y，配置）或 M（m，配置为可加载模块）来设定一个子项。一切设置好之后，选择“<Exit>”退出主菜单。若之前没有选择“<Save>”保存，则会提示用户是否保存设置。用户可以选择“Yes”保存设置，也可选择“No”放弃设置。若保存了设置，然后就可根据新配置生成新内核了。

在配置过程中，用户应谨慎选择，若配置不当可能造成新内核无法启动或行为怪异。还应该注意，不能往内核中配置太多内容，尤其是不使用的内容，否则可能会使内核过大。建议不常用的功能不要直接加入内核，可以采用编译为模块的办法，使用时动态加载。

7.7.2　编译与安装新内核

内核配置完毕后就可编译并生成新内核了，具体步骤如下。

1．运行 make 或者 make all

运行

```
#make                    #或
#make all
```

使用配置文件编译并生成内核及相关文件。这是一个漫长的过程，且需要大量的磁盘空间。

2．运行 make modules_install

执行

```
#make modules_install
```

编译和安装内核模块。安装位置为/lib/modules/$(KERNELRELEASE)。$(KERNELRELEASE)为新安装内核的 release 号。

3．运行 make install

运行

```
#make install
```

安装新内核并配置 grub 启动配置。

4．清理临时文件

由于内核编译过程中生成了大量*.o 等文件，占用了大量空间，内核编译和配置完成后应该将它们清除，腾出空间来供日后使用，方法是执行下列命令之一。

```
#make clean          #清除生成的临时文件，但保留配置和配置支持文件
#make mrproper       #清除生成的临时文件、配置文件和备份文件
#make distclean      #在 mrproper 的基础上再清除备份和补丁文件
```

7.7.3　模块管理

Linux 系统用于内核模块管理的命令有 lsmod、modinfo、insmod、rmmod、modprobe 和 depmod 等。

1．列出已经加载的内核模块（lsmod）

lsmod 用于显示当前内核已经加载的模块信息，方法是：

```
#lsmod
```

直接通过查看/proc/modules 文件的内容也能看到已经加载的模块信息，方法是：

```
#cat /proc/modules
```

2．显示指定模块信息（modinfo）

modinfo 用于显示指定模块本身的属性信息，其用法为：

```
modinfo [-F field] [modulename|filename...]
```

-F field 用于显示模块属性中的指定域；modulename 为模块名；filename 为模块文件名。使用示例如下：

```
#modinfo isofs                     #显示 isofs 模块的所有信息
#modinfo -F filename isofs         #显示 isofs 模块的模块文件名信息
```

3．模块的加载与卸载（insmod、rmmod 和 modprobe）

模块的加载与卸载可由命令 insmod、rmmod 和 modprobe 来实施。insmod 的功能是加载模块；rmmod 的功能与 insmod 相反，用于卸载模块；modprobe 既可用于加载模块，也可用

于卸载模块。它们的用法为：

```
insmod [filename] [module options...]
rmmod [-f] [modulename]
modprobe [-nr] [modulename] [module parameters...]
```

选项-f 用于强制卸载模块；-r 用于模块卸载；-n 用于模拟加载或卸载并显示过程信息，并执行真正的动作。需要说明的是，当用于模块加载或卸载时，modprobe 要优于 insmod 和 rmmod，因为 modprobe 能够处理模块间的依赖关系，而 insmod 和 rmmod 仅简单地执行命令而不考虑后果。使用示例如下：

```
#modprobe wacom                    #加载模块 wacom
#modprobe -r wacom                 #卸载模块 wacom
```

7.7.4　内核参数在线调整

内核参数可以通过配置的方法静态调整，也可在系统运行时进行一定程度的动态调整。用于内核参数动态调整的命令是 sysctl，其用法为：

```
#sysctl [options] [variable[=value]] [...]
#sysctl -p [file or regexp] [...]
```

sysctl 命令依赖于文件系统/proc，系统运行时可调整的参数在/proc/sys/目录内，可以通过 sysctl 对这些参数的读写而达到调整参数的目的。用 sysctl 调整的参数是临时的，系统重启时被重置回原来值。

sysctl 的部分参数如表 7-11 所示。

表 7-11　sysctl 的部分参数

参　　数	意　　义	参　　数	意　　义
-a / -A	显示所有可用参数	-e	忽略未知键错误
-n	显示时不显示变量名，而只显示变量值	-N	显示时不显示变量的值，而只显示变量名
-w	写变量的值	-p <file>	指定配置文件，默认为/etc/sysctl.conf

sysctl 命令的应用示例如下。

```
#sysctl -a                                #显示所有 sysctl 可用参数
#sysctl kernel.hostname                   #显示主机名，等同于命令：uname -n
#sysctl -w kernel.domainname=`uname -n`   #修改域名使之与主机相同
#sysctl -w net.ipv4.ip_forward=1          #配置 IPv4 IP 转发
##说明：对于表示是/非的开关变量，0 表示非，1 表示是
```

习题 7

1．思考题

（1）Linux 系统需要哪些分区？在安装 Linux 系统时应如何规划分区？

（2）Linux 系统的引导程序是 Grub，在系统安装时可安装在哪些位置？效果有何不同？
（3）如何升级你的 Linux 系统？
（4）Linux 系统有哪些常见日志，如何阅读它们？

2．单项选择题

（1）显示系统主机名的命令是（　　）。
A．uname -r　　B．who am I　　C．uname -n　　D．whoami
（2）命令（　　）和（　　）可以显示系统的启动时间和到现在所经历的时间，以及系统内当前用户数信息。
A．uptime　　B．vmstat　　C．uname　　D．w
（3）在红帽系统中，查询已安装软件包 tcsh 内所含文件列表信息的命令是（　　）。
A．rpm -qa tcsh　　B．rpm -ql tcsh　　C．rpm -qp tcsh　　D．rpm -qf tcsh
（4）在 Ubunt 系统中，查询已安装软件包 tcsh 内所含文件列表信息的命令是（　　）。
A．dpkg -l tcsh　　B．dpkg -L tcsh　　C．dpkg -S tcsh　　D．dpkg -s tcsh
（5）在红帽系统中，查询文件/usr/bin/shred 归属某个已安装软件包的命令是（　　）。
A．rpm -qa /usr/bin/shred　　B．rpm -ql /usr/bin/shred
C．rpm -qp /usr/bin/shred　　D．rpm -qf /usr/bin/shred
（6）在 Ubunt 系统中，查询文件/usr/bin/shred 归属某个已安装软件包的命令是（　　）。
A．dpkg -l /usr/bin/shred　　B．dpkg -L /usr/bin/shred
C．dpkg -S /usr/bin/shred　　D．dpkg -s /usr/bin/shred
（7）用于查询试图登录系统但未成功的用户信息的命令是（　　）。
A．last　　B．who　　C．lastb　　D．whoami

3．综合题

（1）系统管理的任务是什么？
（2）sysstat 软件包是传统 UNIX 系统用于系统、磁盘、网络和其他 I/O 的监视工具，它提供有 iostat 和 sar 等实用程序。请安装此包并写出安装的操作步骤。
（3）sar-r 命令用于监视系统的内存和交换区的使用情况，若内存利用率过高或交换设备使用率也较高，则说明了什么问题？
（4）试述编译新内核的方法和步骤。

实验 7

1．系统安装、升级与扩充。
2．临时修改启动参数，让系统启动到 runlevel3.target。
3．修改系统默认启动级别到 runlevel3.target。若需要图形界面，再将默认启动级别设置为 runlevel5.target 或 graphical.target。
4．安装新内核代码。
5．阅读系统常见日志。
6．监视系统活动情况。

第 8 章　设备管理

8.1　设备管理概述

设备管理是操作系统的主要功能之一。在计算机系统中，除了 CPU 和内存等外，其他的大部分硬设备称为外部设备，简称外设。外设包括常用的 I/O 设备、外存设备及终端设备等。这些设备种类繁多，特性各异，操作方式也有很大区别。

设备管理，从系统外部看表现在设备的安装和设备使用上。设备安装可以分两部分，一是设备驱动程序安装，另一是设备的安装与配置。Linux 系统支持即插即用，使设备管理简单多了。

8.1.1　Linux 系统支持的设备

在 UNIX/Linux 系统中，设备文件统一存放在/dev 目录，用户可以通过/dev 目录下的设备文件使用设备。

在早期的 Linux 版本中，/dev 目录内设备文件是静态创建的，包含了所有可能出现的设备文件，内容非常多，但可能这些设备大都不真正存在。

自从 Linux 2.6 内核引入 udev，udev 已经成为当前 Linux 的动态设备管理工具。现在，借助 udev 只为那些已经连接到 Linux 系统的设备生成设备文件，或者说只有被内核检测的设备，才给它创建设备文件；相反，当一个设备被拔掉后，还要删除其设备文件。从这个意义上讲，/dev/目录下的所有文件都有对应的已经连接到系统的真实设备，内容比静态产生时少多了。

Linux 系统支持的常用设备及设备文件如表 8-1 所示。

表 8-1　Linux 系统支持的常用设备及设备文件

设　备　名	设 备 描 述	其 他 说 明
/dev/mouse	鼠标设备	当前鼠标设备
/dev/cdrom,/dev/sr0	CDROM 设备	当前 CDROM 设备
/dev/cdwriter,/dev/sg0	刻录设备	当前刻录设备
/dev/scanner	扫描仪	当前扫描仪设备
/dev/modem	MODEM	当前调制解调器
/dev/root	根设备	根文件系统
/dev/null	空设备	相当于宇宙的黑洞
/dev/zero	0 字符生成设备	这里的 0 字符为'\0'而非 0x30
/dev/pts/#	伪终端设备，用于远程登录	#∈(0,1,…)，会话建立时产生
/dev/usb/*	系统使用 USB 设备	不包括 USB 存储设备
/dev/random	随机数生成器	高质量随机数生成器（当不能产生时将被阻塞）
/dev/urandom	非阻塞式随机数生成器	非阻塞式。当不能产生时，给出伪随机数
/dev/fdX*	软驱设备	X=0，…，取决于具体的软驱个数，默认为/dev/fd0
/dev/sdX	SCSI 硬盘设备	X=a，b，…，取决于具体硬盘个数

（续表）

设　备　名	设 备 描 述	其 他 说 明
/dev/sdXY	SCSI 硬盘设备上的分区	X 同上，Y=1，…，取决于具体分区个数
/dev/hdX	IDE 硬盘设备，整个硬盘或光驱	X=a，b，…，取决于具体硬盘个数
/dev/hdXY	IDE 硬盘设备上的分区	X 同上，Y=1，…，取决于具体分区个数
/dev/tty#	终端设备	#∈(0，1，…)，数量取决于真实终端的个数
/dev/ttyS*	串口通信设备	数量取决于真实串口的个数
/dev/console	系统主控台	单用户或系统维护模式时使用
/dev/lpX	并口设备，用于打印机	X∈(0，1，…)，用于并口打印机
/dev/loopX	loopback 设备，用于安装映像文件	X∈(0，1，…)，可根据需要自动生成
/dev/loop-control	Loopback 设备的控制设备	
/dev/st*	SCSI 磁带机，可自动倒带	取决于真实配置
/dev/nst*	SCSI 磁带机，不自动倒带	取决于真实配置
/dev/ht*	IDE 磁带机，可自动倒带	取决于真实配置
/dev/nht*	IDE 磁带机，不自动倒带	取决于真实配置
/dev/scdX	SCSI 光驱	X=0，1，…，取决于真实数量

表 8-1 中只列出了很少一部分常见设备，用户从/dev/目录也不能看到 Linux 系统所支持的全部设备，欲了解更多的设备及设备文件相关信息，可参见内核源代码包中的 Documentation /devices.txt 文件（参见 7.7 节）。

8.1.2　设备发现与 udev 简介

udev 是一种动态管理/dev 目录的工具，但它需要依赖于被安装在/sys 的 sysfs 文件系统。

在 Linux2.6 内核引入 udev 的同时，也引入了 sysfs。内核直接管理 sysfs，并导出当前系统中已经连接的设备基本信息到 sysfs。一旦有设备连接到系统，当被内核监测到时，就直接将其注册到 sysfs，并通知 udev 去检测/sys 中属于该新设备的信息，然后由 udev 为之在/dev/创建设备文件。

udev 只是一个框架，它的行为完全受它的规则所控制。udev 规则文件以.rules 作为扩展名。udev 根据规则文件为匹配规则的设备创建或删除设备文件或文件链接。规则文件被安排在/etc/udev/rules.d、/run/udev/rules.d 和/lib/udev/rules.d 三个不同的地方。udev 从以上目录读取规则文件，并按字典排列法排序。当有同名文件时，相互覆盖，级别高低依次是/etc、/run 和/lib 目录中的文件。

udev 的配置文件为/etc/udev/udev.conf，它的作用是让用户修改 udev 的默认参数值。在本书中所涉及的系统中，udev 均是使用默认值工作的，故不需要太多考虑此文件。

udev 的工作过程是：当从内核收到一个设备变动事件时，判断添加或删除设备。若是设备删除，则删除/dev/目录内相应的设备文件名；若是添加设备，则从配置文件/etc/udev.conf 中查找规则文件及位置，然后在规则文件目录内依次查找所有规则文件，若有与本次设备相匹配的规则，则在/dev/目录下为设备创建设备文件，否则在/dev/目录下以内核提供的设备名为设备名创建设备文件。

8.1.3　系统设置与查看

1．查看硬件信息

1）查询系统启动日志（dmesg）

查看系统启动日志，可以了解系统中连接的所有硬件设备情况（参见 7.5.2 节），但输出的信息比较原始，需要仔细甄别。

2）查询 PCI 总线和所连设备信息（lspci）

lspci 的功能是查询系统中的所有 PCI 总线和连接的设备信息，其用法为：

```
lspci [options]
```

在默认情况下，lspci 仅显示简洁的信息，但可以通过选项让其显示更多的信息。常用选项有：-v、-vv、-vvv 用于显示更多的信息；-n、-nn 用于显示厂家和设备代码信息；-m、-mm 用于显示设备数据信息；-t 用于树状显示。示例如下：

```
#lspci                          #查询所有 PCI 设备信息
#lspci -v                       #输出更多的信息
#lspci | grep -i ethernet       #查询网卡信息
#lspci | grep storage           #查询存储设备信息
```

3）查询 USB 总线和所连接设备信息（lsusb）

lsusb 的功能是查询系统中的所有 USB 总线和所连接的设备信息，其用法为：

```
lsusb [options]
```

lsusb 通常以简洁方式显示信息，选项-v 让其显示更多信息；-t 用于树状结构显示。示例如下：

```
#lsusb                          #显示 USB 总线和所连接的设备信息
#lsusb -t                       #树状格式显示
```

4）查询 CPU 信息（lscpu）

lscpu 用于显示有关 CPU 的体系结构等信息，其用法为：

```
lscpu [-a|-b|-c] [-x] [-e[=list]|-p[=list]]
```

lscpu 从/proc/cpuinfo 搜集信息，可使用选项改变默认输出行为。选项-e、--extended[=list]以便于阅读的方式显示；-p，--parse[=list]以优化方式输出；-b、-c 和-a 分别用于显示在线、离线和全部 CPU 信息，但要配合-e 或-p 使用。

lscpu 的输出信息包括 CPU-（逻辑数）；CORE（逻辑核数）；SOCKET（逻辑 socket 号）；BOOK（逻辑 book 号）；NODE（逻辑 NUMA node 数）；CACHE（CPU 间的 CACHE 信息）；ADDRESS（CPU 物理地址）；ONLINE（CPU 使用情况）；CONFIGURED（是否支持动态分配）；POLARIZATION（虚拟模式下能否改变分配方式）；MAXMHZ（最大 MHZ）和 MINMHZ（最小 MHZ）。-e 和-p 选项中的 list 就是这些由逗号“,”的连接列。示例如下：

```
#lscpu                          #显示所有 CPU 信息
#lscpu -e                       #以易读方式显示
#lscpu -p=cpu,core,SOCKET       #只显示 cpu、core 和 SOCKET 信息
```

2．查看块设备信息

1）块设备查询（lsblk）

lsblk 的功能是查询所有或指定块设备信息，其用法为：

```
lsblk [options] [device...]
```

常用选项有：-a/--all 输出所有设备；-d/--nodeps 只显示主设备；-f/--fs 显示文件系统类型；-l/--list 以表方式显示（默认以树状方式显示）；-O/--output-all 输出所有列；-o,/--output +list 输出指定列；-S/--scsi 只输出 SCSI 设备。

示例如下：

```
#lsblk                               #显示所有块设备
#lsblk -S                            #只显示 SCSI 设备
#lsblk /dev/sda                      #只输出 sda 信息
#lsblk -l -o"NAME,LABEL,UUID"        #输出设备的 NAME、LABEL 和 UUID 信息
```

2）显示块设备属性（blkid）

blkid 用于查找和显示块设备属性信息，其简单用法为：

```
blkid [-L label | -U uuid] [device]
```

选项-L label 用于显示 LABEL 为 label 的块设备属性信息；-U uuid 用于显示 UUID 为 uuid 的块设备属性信息；device 为设备名，可以是/dev/中的块设备名，也可以是/dev/disk 下 by-id、by-label、by-path 和 by-uuid 中的设备名。当不带任何参数执行 blkid 时，将显示所有块设备的属性信息。示例如下：

```
#blkid                          #显示所有块设备信息
#blkid /dev/sda                 #显示 sda 设备信息
```

3．查看网卡信息（nmcli）

nmcli 是一个控制 NetworkManager 和报告网络状态的命令行工具，可以用来创建、显示、编辑、删除、启用和停用网络连接，还能控制和显示网络设备状态。

nmcli 需要软件包 NetworkManager（红帽系统）或 network-manager（ubuntu）的支持，且要启动 NetworkManager.service。使用前需要安装所需软件包并激活和启动 NetworkManager.service。

1）用法及选项

nmcli 的用法为：

```
nmcli [options] obj { cmd | help }
```

nmcli 的常用选项有：-t 简洁输出；-p 漂亮输出；-m tabular|multiline 单行 Tab 分隔或多行输出；-c auto|yes|no 颜色自动、彩色或黑白；-f <field1,...>|all|common 输出指定域、所有域

或通用域；-h 帮助。

2）操作对象 obj 及子命令

nmcli 的操作对象 obj 及相关子命令如下所述。

g[eneral]：一般状态和操作，可用子命令有 status | hostname [<hostname>]| permissions | logging [level level] [domain domains...]。

n[etworking]：网络控制，可用子命令有 on | off | connectivity [check]。

r[adio]：无线广播控制命令，可用子命令有{all | wifi | wwan} [on | off]。

c[onnection]：连接管理，可用子命令有{show | up | down | add | edit | modify | delete |monitor | reload | load} [arg ...]。

d[evice]：网络设备管理，子命令有{status | show | set | connect | reapply | disconnect |delete | monitor | wifi | lldp} [arg ...]。

a[gent]：secret 或 polkit 代理。

m[onitor]：活动监视器，参见 nmcli connection monitor 和 nmcli device monitor。

3）使用示例

nmcli 功能强大，这里只介绍其网络接口管理方面的示例，更多用于网络管理的示例参考 nmcli-examples 的 man 手册。

（1）显示一般信息

```
#nmcli g                          #显示状态信息
#nmcli g hostname                 #查询主机名
#nmcli g hostname fedora24        #修改系统主机名
```

（2）显示设备信息

```
#nmcli device    #查询设备信息（包括 DEVICE、TYPE、STATE 和 CONNECTION）
    DEVICE    TYPE       STATE        CONNECTION
    eth0      ethernet   connected    eth0
    lo        loopback   unmanaged    --
```

（3）显示连接信息

```
#nmcli conn show     #查询网络连接（包括 NAME、UUID、TYPE 和 DEVICE）
    NAME     UUID                                   TYPE             DEVICE
    eth0     925afd2d-3089-3071-bd06-e86920f9fd08   802-3-ethernet   eth0
    virbr0   3415258b-a64a-4e91-bfb5-7e8c27cc84d3   bridge           virbr0
```

8.2 打印机的管理与使用

UNIX/Linux 系统使用 CUPS（Common UNIX Printing System）作为默认的打印机管理程序。

CUPS 提供了强大的打印机管理功能，可在各种常用的 UNIX 版本中使用，它能够支持大多数的常用型号打印机。

在 Linux 的发布盘上带有 CUPS 系统软件包，用户可在系统安装时选择安装，也可在安装之后的系统扩充时安装。通常，在系统安装时会自动安装该软件包。

8.2.1 cups 的安装

与 CUPS 相关的软件包为 cups，若系统要使用打印服务，需要安装它，方法是：

```
#yum install cups              #红帽系列
#apt-get install cups          #ubuntu 系统
```

8.2.2 打印机安装与配置

在 cups 软件包被安装以后，还必须对打印机进行安装和配置才能使用。对打印机的管理内容有添加打印机和对现有打印机的管理。

1．图形界面

1）添加本地打印机

首先将打印机连接到系统，打开电源，若能自动被识别，则配置结束；否则按以下方式执行打印机设置程序

```
Setting→Printers               #首先启动设置中心
system-config-printer          #CLI 界面执行
```

进入打印机配置主界面（如图 8-1 所示），通过此界面可以配置已经存在的打印机，也可以增加新打印机。

如果要新建打印机，选择“新建”选项，出现“New Printer（新打印机）”的“连接选择”界面（如图 8-2 所示），可以选择本地并口、本地串口、网络打印机等，这里选择本地并口（LPT#1），选择后单击“前进”选项，出现“打印机品牌/厂商选择”界面（如图 8-3 所示），选择后单击“前进”选项，进入“型号”选择界面（如图 8-4 所示），选择型号后单击“前进”选项，出现“打印机名”、“描述”和“位置”设置界面（如图 8-5 所示），设置完毕后，单击应用完成打印机新建过程。接着是打印测试页。在打印机新建过程中，可能会因系统所连接的物理打印机不同而有所区别，要根据实际情况进行操作。

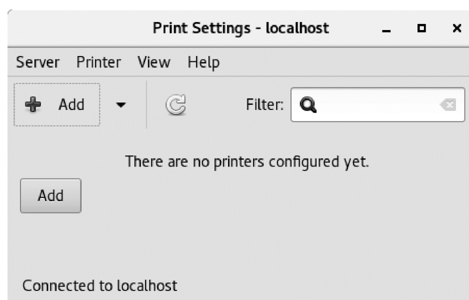


图 8-1　打印机管理主界面

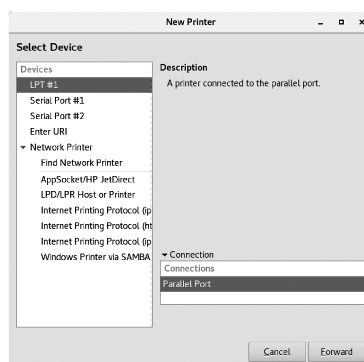


图 8-2　连接选择界面



图 8-3　新打印机选择图

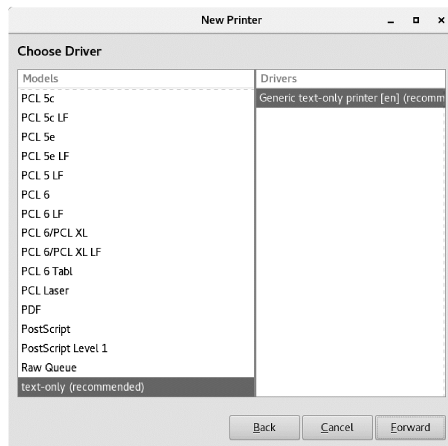


图 8-4　型号选择界面

在如图 8-5 所示的界面中，“Printer Name（打印机名）”是逻辑名，可由用户随便指定，比如 lp0、prn1 等。“Description（描述信息）”也由用户根据打印机情况来填写，比如厂家、型号等，也可不填。“Location（位置）”为打印机所在的主机或 IP 地址，也可以不填。

2）对现有打印机的配置

对现有的打印机，在选中后可以右击之，在弹出的（如图 8-6 所示）快捷菜单中，勾选“Enabled”启用或禁用它，勾选“Shared”共享它；也可以选择“Delete”删除之，或选择“Rename”为其更名，如果选择“Properties”则可以对其进行修改。

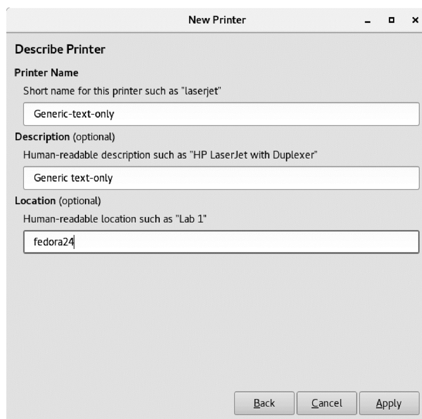


图 8-5　打印机名和位置设置界面

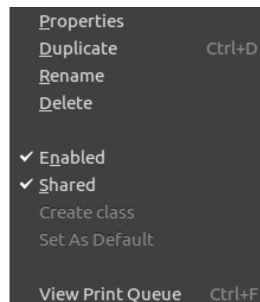


图 8-6　打印机快捷菜单

在选择“Properties”之后，将出现如图 8-7 所示的打印机属性界面，用户可以在左侧的功能栏中选择“Settings（设置）”、“Politics（策略）”、“Access Control（访问控制）”、“Job Options（作业选项）”和“Ink/Toner Levels（墨水/色粉）”进行设置。

通过“Setting（设置）”可以设置“Description（描述）”，所在主机“Location（位置）”，“Device URI（设备 URI）”，“Make and Model（厂家和型号）”、“Printer State（打印机状态）”和“打印测试页（Print Test Page）”。

通过如图 8-8 所示的“Politics（策略）”设置界面，可以勾选“Enabled（启用）”打印机，并允许勾选“Acceptjobs（接收任务）”和“Shared（设为共享）”，让打印机共享并允许接收

打印任务；在“Politics 栏”可以设置“错误”和“操作”策略；在“Banner”栏可设置“开始”和“结束”Banner。对于“开始”和“结束”Banner 建议不设置。若设置了，将在打印正文页之前输出开始 Banner（前封面），之后输出“结束”Banner（底封面），这无疑是对纸张的浪费。但是，若系统是用于打印服务共享服务器的，就应该配置 Banner 了，因为这样可以区分打印件的归属。

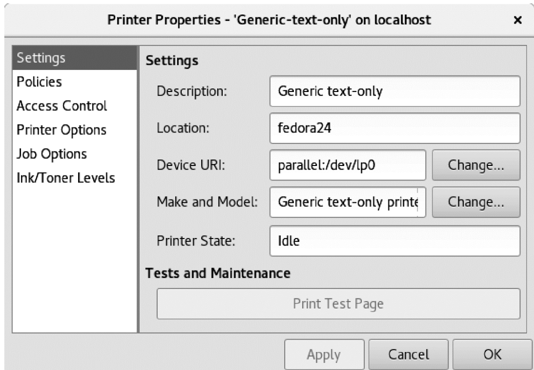


图 8-7　打印机属性界面

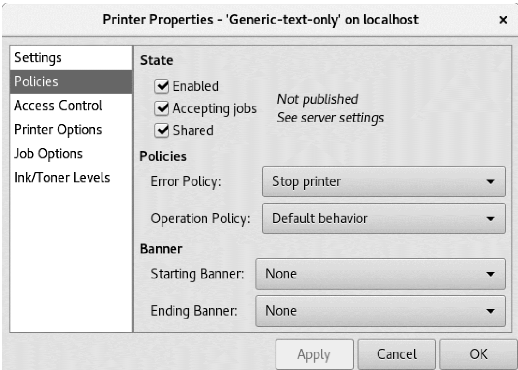


图 8-8　打印机 Politics（策略）设置界面

通过“Access Control（访问控制）”可以设置“Allow …（允许）”和“Deny …（不允许）”使用该打印机的用户。

通过“Job Options（作业选项）”可以设置“Copies（份数）”、“Orientation（纸张方向）”、“Pages per Side（每面页数）”。

通过“Ink/Toner level”可以设置打印质量。

2．通过 Web 页面管理打印机

CUPS 支持浏览器风格的配置与管理，在所有桌面或工作站系统上，均可以在浏览器界面下进行配置，方法是在浏览器地址栏输入 http://localhost:631/，会出现如图 8-9 所示的基于 Web 的打印机管理界面。

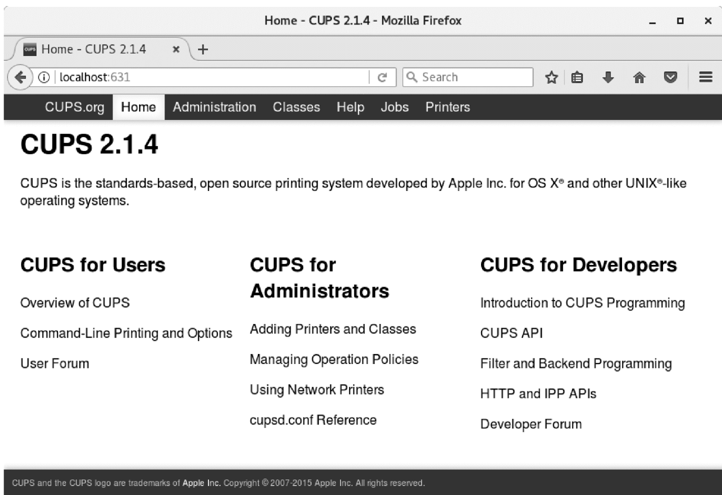


图 8-9　打印机管理界面

在该界面中，可选择“Administration”进行打印机配置，选择“Jobs”可进行打印任务管理，选择“Printers”进行打印机管理。

3．通过字符界面管理打印

关于打印机的管理和使用还有一套字符界面的命令，可供没有图形界面的 server 版系统使用，这些命令及功能描述如表 8-2 所示，本书将在稍后只介绍 lpr/lp、lpstat 和 cancel/lprm 的使用，其他命令留给读者自行研究。

表 8-2　打印机管理字符界面命令

命　　令	功 能 简 述
accept/cupsaccept	允许指定的目标接收打印任务
cancel/lprm	取消打印任务
cupsdisable	禁用指定的打印机或类
cupsenable	启用指定的打印机或类
lpadmin	设置或更改打印机或类的配置
lpinfo	显示 CUPS 服务器已知的可用设备或驱动程序
lpoptions	显示或设置打印机选项和默认值
lpq	显示当前打印队列状态
lpr/lp	提交打印任务
lpstat	显示打印队列和请求状态信息
reject/cupsreject	拒绝指定的目标接收打印任务

8.2.3　cups 系统的启动管理

在 Linux 系统中 cups 是独立服务，服务名为 cups.service。可设置为开机自启动来为用户提供服务。

1．立即启动、停止或重启动 cups 服务

立即启动、停止或重启动 cups，可以通过以下命令：

```
#systemctl start cups.service           #立即启动
#systemctl restart cups.service         #重新启动
#systemctl stop cups.service            #停止
#systemctl status cups.service          #检查 cups 服务状态
```

2．配置系统启动时自动启动 cups 服务

配置系统启动时自动启动或不启动 cups，可以通过以下命令：

```
#systemctl enable cups.service          #设为开机自启动
#systemctl disable cups.service         #设为开机不启动
```

8.2.4　打印机使用

1．图形界面

在桌面系统中，用户可通过以下方法使用打印机：

（1）在应用程序中按顺序选择“文件”→“打印”。

（2）单击“打印机”图标。

2．字符界面

命令方式下用于文件打印的命令是 lp 或 lpr，分别简介如下。

1）lp 命令

lp 的常用方法为：

```
lp [-d dest_printer] [-h server_name] [-n copies] [-P page_lists] [files]
```

参数说明参见表 8-3。

表 8-3　lp 命令的部分参数

参　数	意　义
-d dest_printer	指定打印机，若不指定则使用默认打印机
-h server_name	指定打印机位置或服务器，若不指定则默认为本机
-n copies	指定打印份数，若不指定则只打印 1 份
-P page_lists	指定文件的打印范围，若不指定则为全部文档。指定时可使用页号或页的范围，例如，1，4～6，10
files	指定打印的文件，若不指定则使用标准输入

2）lpr 命令

lpr 的常用方法为：

```
lpr [-P dest_printer] [-H server_name] [-#copies] [files]
```

选项-P 用于指定打印机，-H 用于指定服务器，-#用于指定打印份数。

3）打印示例

```
#lp myp.c                  #打印文件 myp.c
#lp -n 2 myp.c             #打印 2 份文件 myp.c
#lpr -#2 myp.c             #同上
#ls -l /tmp | lp           #打印某个目录的列表，如/tmp 目录
#lpr file.c file2.txt      #打印多个文件
#pr -n sh.c | lpr          #以文件名为标题，并添加行号分页打印 c 语言文件 sh.c
```

8.2.5　打印任务管理

1．图形界面

当打印队列中有打印任务时，会在桌面的状态区出现打印机图标，单击该图标可以打开如图 8-10 所示的打印队列管理界面。选中打印任务后，用户可以通过界面上面的按钮或右击该打印任务，在弹出的（如图 8-11 所示）快捷菜单中可选择“Cancel”、“Hold”、“Release”和“Reprint”而撤销、暂停、释放或重新打印它。

图 8-10　打印机队列管理界面

图 8-11　打印任务操作菜单

2．字符界面

在字符界面下，可通过 lpstat、cancel 和 lprm 等命令对打印队列进行管理。

1）打印队列查询（lpstat）

lpstat 命令用于打印机队列的查询，其常用方法为：

```
lpstat [ -a [printers]] [-d] [-l] [-o [printers]]
[-p [printers]] [-r] [-s] [-t] [-u [users]] [-v [printers]
```

lpstat 所使用的部分参数如表 8-4 所示。

表 8-4　lpstat 命令常用参数

参　　数	描　　述
-a [printers]	显示指定打印机的接收状态，若不指定则显示所有打印机
-d	显示默认打印机
-l	显示打印机、队列等列表
-h server	指定服务器名
-o [printers]	显示指定打印机上的作业队列。若不指定则显示所有队列
-p [printers]	显示指定打印机是否激活，若不指定则显示所有打印机
-r	显示 cups 服务是否启动
-s	显示打印机状态汇总信息，相当于同时使用-d，-c 和-p
-t	显示所有状态信息，相当于同时使用-r，-d，-c，-v，-a，-p 和-o
-u [users]	显示指定用户的作业队列，若不指定用户则为当前用户
-v [printers]	显示指定打印机所使用的设备文件

2）撤销打印任务（cancel、lprm）

cancel、lprm 用于撤销打印队列中的作业，常用方法为：

```
cancel [-a] lp_job …
lprm lp_job_num …
```

选项-a 用于取消队列中的所有打印作业。

3）使用示例

```
#lpstat -o                              #首先显示打印队列
    lp-4        root        *
    lp-5        gjshao      *
#cancel lp-5 lp-4                       #删除打印作业 lp-5 lp-4
#lprm 5 4                               #同上
#cancel -a                              #取消所有打印作业
```

8.2.6 cups 的配置文件

cups 系统配置目录为/etc/cups，在此目录内存放着 cups 的配置文件，通过不同的配置文件可以对相应的功能进行配置。这些配置文件分别是：

服务器配置文件/etc/cups/cupsd.conf

客户端配置文件/etc/cups/client.conf

打印机配置文件/etc/cups/printers.conf

类配置文件/etc/cups/classes.conf

8.3 交换区管理

8.3.1 概述

几乎所有多任务系统都需要交换区，用于对内存的逻辑扩充。由于物理内存的容量有限，而装入的程序是随机的，内存可能不能全部装入所有正在运行的程序，系统往往是将那些正在运行的程序放在内存，而将那些处于阻塞或等待的进程放到外存交换区上。

所有的 UNIX 操作系统都有对交换区 swap 的要求。不论在安装过程中是否指定了交换区，系统总要分配一定大小的磁盘空间用作交换区或交换设备。

交换区由操作系统自动管理，一般不需要用户或管理员进行过多的干预。但系统管理人员可通过特殊命令对它进行一定范围的控制或配置。

如果在系统安装时没有分配足够的交换区，对于正在使用的系统，管理员可以使用交换区管理命令添加交换区。若外存（主要指硬盘）上有剩余空间则可创建交换区，此时建议使用独立的分区，若不能创建交换分区的话，也可使用文件作为交换设备。

用于交换设备管理的命令有 mkswap，swapon，swapoff 等。

1．mkswap

mkswap 的功能是在设备或普通文件上创建交换区，其用法为：

```
mkswap [-c] [-vN] [-f] [-p PSZ] [-L label] device [size]
```

device 为交换区文件名，可以是设备文件也可以是普通文件；参数-c 用于在创建时做坏块检查，其他参数可以忽略。-v0 和-v1 分别用于创建旧式或新式的交换区，默认为后者；-f 用于强制创建；-p PSZ 用于指定页面大小；-L label 用于指定标签，允许使用标签加载；size 用于指定大小，一般不指定。

若在某个设备上构造交换区，则设备将作为裸设备使用，其上的文件系统将被破坏。

mkswap 拒绝创建小于 10 个页面的交换设备。查询系统页面大小的方法有很多，最直接办法是由命令 getconf PAGESIZE 获取。

2．swapon

swapon 的功能启用由 mkswap 创建的交换区，其常用方法为：

```
/sbin/swapon [-h -V]
/sbin/swapon -a [-v] [-e]
/sbin/swapon [-v] [-p priority]   swap_nam ...
/sbin/swapon [-s]
```

swap_name 为由 mkswap 创建的交换设备文件。-a 用于启用/etc/fstab 文件内的所有交换设备；-e 与-a 配合使用时默认忽略不存在的交换设备；-s 用于显示系统交换设备的使用情况；-L label 加载具有标签 label 的交换设备；-p priority 指定加载优先级；-v 用于显示冗余信息；-V 显示版本号。

3．swapoff

swapoff 的功能用于卸载交换区，其用法为：

```
swapoff [-ahV] swap_nam ...
```

参数意义与 swapon 相同。

8.3.2　使用交换设备

交换设备的使用方法是首先使用 mkswap 命令在独立分区上创建交换区，然后再使用 swapon 命令激活它。如果要让系统在启动时自动启用交换区，则需在文件/etc/fstab 中增加一行对交换区管理的内容。

设有一个独立分区/dev/sda9，要将它用作系统的交换设备，则可按以下步骤进行。

（1）创建分区：

```
mkswap -c /dev/sda9
```

（2）激活分区：

```
swapon /dev/sda9
```

（3）在文件/etc/fstab 中增加如下行，让系统启动时自动加载交换设备。

```
/dev/sda9 none swap defaults 0 0
```

8.3.3　使用交换文件

当外存上无独立分区时，也可使用文件作为交换设备。使用交换文件作为交换区时，必须首先创建一个指定大小的文件，此项工作可由 dd 来完成。

创建一个大小为 20MB 的文件并用作交换设备的示例如下。

（1）创建一个大小为 20MB 的文件/usr/swap_add：

```
dd if=/dev/zero of=/usr/swap_add bs=1M count=20
```

（2）构造交换设备：

```
mkswap /usr/swap_add
```

（3）激活它供系统使用：

```
swapon /usr/swap_add
```

8.4　串口的管理与使用

串行通信口（Serial Communication Port，简称串口）在计算机和操作系统的发展过程中起了至关重要的作用，在以太等高速网卡出现之前，大多的联网或通信工作都是由串口完成的。随着计算机技术的发展，现在很多微机都不配置串口了，串口应用越来越显得不那么重要了，但事实上串口还在计算机通信中起一定作用。在 Linux 系统中，串口支持的网络通信协议有 PPP 和 SLIP 等。

8.4.1　Linux 系统的串口设备

在 DOS/Windows 系统使用串口时的设备名称为 COM1:，COM2:，COM3:，COM4:，而在 Linux 系统的命名方法与 DOS/Windows 不同，所对应的设备名称依次为 ttyS0、ttyS1、ttyS2 和 ttys3。在一些旧系统中还保留另一种命名方式，串口依次命名为 cua0、cua1、cua2 和 cua3，分别代表串口上的拨出设备。

串口总是与中断（IRQ）和 I/O 端口联系着，两者的组合定义一个串行通信口。在默认情况下，COM1 和 COM3 使用相同的中断和端口，COM2 和 COM4 使用相同的中断和端口（参见表 8-5）。

表 8-5　串口名称及中断和端口表

Linux 设备	DOS/Windows 设备	中断（IRQ）	端口（Port）	串口拨出设备
/dev/ttyS0	COM1:	4	3F8	/dev/cua0

（续表）

Linux 设备	DOS/Windows 设备	中断（IRQ）	端口（Port）	串口拨出设备
/dev/ttyS1	COM2:	3	2F8	/dev/cua1
/dev/ttyS2	COM3:	4	3F8	/dev/cua2
/dev/ttyS3	COM4:	3	2F8	/dev/cua3

由于 AT/ISA 体系结构的限制，多个串口是不能共享中断的，因此设备/dev/ttyS0 和/dev/ttyS2 不能同时工作，同样/dev/ttyS1 与/dev/ttyS3 也不能同时工作。如果需要更多的串口，可以使用多用户卡，从一个串口上分成多个。多用户串行卡有 4 户、8 户、16 户……这样的一个多口卡只占用一个中断，可被多用户同时使用，这是 UNIX/Linux 环境下多用户同时使用系统的基本形式（参见图 8-12）。

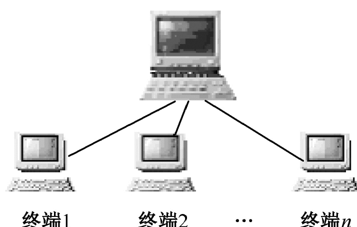


图 8-12　UNIX/Linux 终端多用户

Linux 系统提供多种串口管理软件，以下介绍 setserial 和 minicom 等，需要时先安装它们。

8.4.2　串口管理程序（setserial）

1．功能及用法

setserial 用于 Linux 系统的串口管理，其功能是设置或显示与串口相关的配置信息，其用法为：

```
setserial serial-device -abqvVWz [cmd1 [arg]] ...
setserial -g [-abGv] device1 ...
```

2．命令行参数

setserial 的命令行参数如表 8-6 所示，用于串口设置的主要子命令如表 8-7 所示。

表 8-6　setserial 的命令行选项

参　数	描　述	参　数	描　述
-a	显示详细信息	-v	显示较多信息
-b	显示摘要信息	-G	显示可以用于设置串口格式的串口信息
-q	安静工作，显示较少信息	-z	设置前，先将所有设置项的值设为 0

表 8-7　串口设置的部分子命令

串 口 命 令	描　　述
port <串口编号>	串口编号，参见表 8-5
irq <IRQ 编号>	IRQ 编号，参见表 8-5
uart <UART 类型>	UART（Universal Asynchronous Receive/Transmitter）类型。其值可为：none，8250，16450，16550，16550A，16650，16650V2，16750，16850，16950，16954
autocnfig	让内核自动检测并设置串口
auto_irq	让内核自动检测并设置 IRQ
^auto_irq	配合 autoconfig 使用，关闭 auto_irq 功能
skip_test	配合 autoconfig 使用，关闭自动检测 UART 功能
^skip_test	配合 autoconfig 使用，让内核自动检测 UART 并设置
baud_base	设置基本波特率（CLOCK_FREQ/16）
spd_hi	使用 56Kb/s 而非 38.4Kb/s
spd_vhi	使用 115Kb/s 而非 38.4Kb/s
spd_normal	使用 38.4Kb/s
session_lockout	锁住串口，不让其他用户使用
^session_lockout	解除由 session_lockout 操作的锁定状态
pgrp_lockout	锁住串口不让其他组使用
^pgrp_lockout	解除由 pgrp_lockout 操作的锁定状态

3．使用 setserial 设置串口示例

1）串口设置

```
#setserial /dev/ttyS0 -v autoconfig                   #让系统自动设置串口/dev/ttyS0
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4        #命令输出
#setserial -G /dev/ttyS0                  #以 setserial 命令行参数格式输出/dev/ttyS0 的设置
/dev/ttyS0 uart 16550A port 0x03f8 irq 4 baud_base 115200 spd_normal …      #命令输出
#以与/dev/ttyS0 相同的参数设置/dev/ttyS1
```

2）使用临时变量和变量替换

```
#setserial /dev/ttyS1 `setserial /dev/ttyS0 −G | awk ’{print substr($0,11)}’`       或
#tx=`setserial /dev/ttyS0 −G | awk ’{print substr($0,11)}’`; setserial /dev/ttyS1 $tx
```

8.4.3　串口和调制解调器管理及通信程序（minicom）

minicom 是一个串口和调制解调器管理及通信程序，能够运行于多数 UNIX 系统。它具有以下特性：自动重拨号的拨号目录，对串行设备支持 UUCP 格式的 lock 文件，独立的脚本语言解释器，文件捕获，多用户单独配置等。minicom 的用法为：

```
minicom [-somMlwz8] [-c on|off] [-S script] [-d entry]
        [-a on|off] [-t term] [-p pty] [-C capturefile] [configuration] …
```

minicom 有很多参数，常用命令行参数如表 8-8 所示。

表 8-8　minicom 的常用命令行参数

选　　项	功能及描述
-a on|off	改性使用。有些终端有特性处理属性，-a on|off 用于打开或关闭终端属性
-c on|off	打开或关闭颜色属性
-C capturefile	指定启动时使用捕获文件为 capturefile
-d <entry>	启动或直接拨号
-s	不进行初始化而是直接进入配置菜单。root 使用此选项在/etc/minirc.dfl 中编辑系统范围的默认值
-S script	在启动时，执行指定的 script 文件。如果还使用了-d，意在启动时拨号，此脚本将在拨号之前运行
-o	启动时不初始化调制解调器
-m	以 Alt 或 Meta 键作为指令键
-M	与-m 同，但假定 Meta 键设置字符高端的第 8 位（发送 128+字符代码）
-p pty	使用伪终端 pty
-l	逐字翻译高位被置位的字符
-w	启用自动换行
-z	在终端上显示状态行
-8	不加修改地直接传送 8 位字符
configuration	用于指定配置文件。若不指定则使用/etc/minirc.dfl

当使用命令 minicom -s 启动时，minicom 进入如图 8-13 所示的设置界面。

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

图 8-13　minicom 的设置界面

用户可以在此界面通过“↑”或“↓”选择“Serial port setup”和“Modem and dialing”进行设置，参见图 8-14 和图 8-15。在图 8-14 所示的界面中，用户可以设置串口参数：

A——串口设备，如/dev/ttyS0；

B——锁文件，如/var/lock；

C——接入程序，如 mgetty，uugetty 等；

D——拨出程序，如 chat；

E——设置速率/奇偶校验/校验位，如 38400 8N1；

F——硬件流控，如 Yes；

G——软件流控，如 No。

如果用户不关心接入和拨出程序，在设置时可以不设置。如果设置了接入和拨出程序，则它们必须是属于 root 且具有 suid 属性。

通过如图 8-15 所示界面，可以设置连接在串口上 MODEM 的参数。

若安装有 MODEM，不带任何参数运行 minicom，则 minicom 进入如图 8-16 所示的界面，在此界面按下 Ctrl+A 后，再输入 Z 后出现如图 8-17 所示的菜单界面，选择 O 后同样进入串口设置界面。

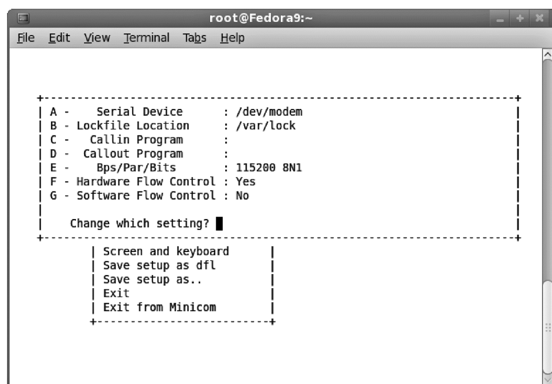


图 8-14　串口设置界面

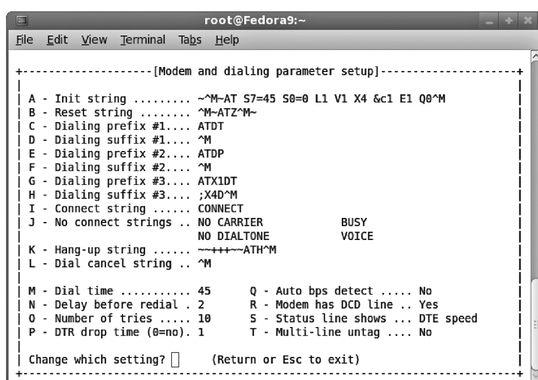


图 8-15　MODEM 设置界面

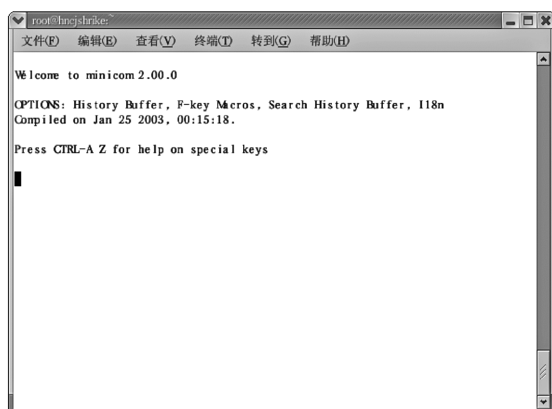


图 8-16　minicom 设置界面

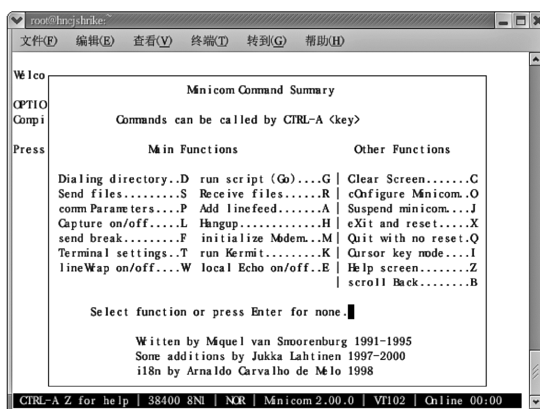


图 8-17　minicom 功能菜单界面

习题 8

1．思考题

（1）/dev 目录的作用是什么？请说出一些常用的设备及其设备文件名。

（2）如何得到一个大小为 1KB 的内容全为'\0'的数据文件 zero_file?

（3）/dev/null 和/dev/zero 的作用是什么，如何使用/dev/null 和/dev/zero 设备？

（4）如何产生随机数？在产生随机数时，/dev/random 和/dev/urandom 有何不同？

2．单项选择题

（1）Linux 系统中用于文件打印的命令是（　　）和（　　）。

A．lp　　B．lprm　　C．lpr　　D．lpstat

（2）Linux 系统中用于打印队列查询的命令是（　　）。

A．lp　　B．lprm　　C．lpr　　D．lpstat

（3）Linux 系统中用于打印作业删除的命令是（　　）和（　　）。

A．lp　　B．lprm　　C．cancel　　D．lpstat

（4）Linux 系统中用于串口设置的常用命令是（　　）和（　　）。

A．setup　　B．setserial　　C．minicom　　D．mknod

（5）Linux 系统中用于串口通信的设备类型是（　　）和（　　）。

A．/dev/ttyS*　　B．/dev/tty[ab]*　　C．/dev/console　　D．/dev/cua*

（6）Linux 系统的默认光驱和刻录设备分别是（　　）和（　　）。

A．/dev/cdrom　　B．/dev/hdc　　C．/dev/cdwriter　　D．/dev/sdc

3．综合题

（1）试述交换区的作用是什么？在系统安装时应该如何估计并配置交换设备的大小？若系统安装时设置的交换区不足，应如何再为系统增加交换空间？

（2）试在系统内以不同用户在多个终端上登录，并执行 w 或 who 命令，观察每个用户使用的终端，再在每个终端（TUI 或 CLI）上执行命令 tty，查询自己使用的终端设备。

实验 8

1．cups 软件包的安装和打印机的使用。

2．交换设备的使用。

3．空设备/dev/null 和 0 字符设备/dev/zero 的使用。

第 9 章　网络配置、管理与基本应用

9.1　TCP/IP 基础知识

世界上有各种不同类型的计算机，它们使用不同的操作系统，要想让这些计算机之间相互通信，就必须有统一的标准，最著名的两个标准分别是 OSI　7 层开放模型和互联网模型 TCP/IP。TCP/IP 是由 TCP 和 IP 等协议所构成的通信协议簇，它规范了网络上所有通信设备之间的数据传送格式和传输方式，是目前 Internet 所采用的通用标准。

TCP/IP 协议采用了 4 层结构，每一层都有特定的用途，并且有多个协议与之相关。这 4 层分别为应用层、传输层、网络层和网络接口层。

9.1.1　IP 地址

为了使众多连入 Internet 的主机在通信时能够相互识别，Internet 给每一台主机都分配了一个唯一的 32 位地址，称为 IP 地址。

1．IP 地址格式

为便于主机的寻址，每个 IP 地址都分为网络号和主机号两部分，其格式如图 9-1 所示。

由于一个 IP 地址由 32 位二进制数构成，难以阅读和记忆，为了使用方便，人们将它分成 4 组，每组 8 位，各组都以十进制表示，并用圆点分开，这种表示方法称为“点分十进制表示法”，例如：

二进制 IP 地址 11001010110001001111000100001000

十六进制表示为 CAC4F008

用点分十进制表示为 202.196.240.8

2．IP 地址的分类

IP 地址分为五类：A 类、B 类、C 类、D 类、E 类，其中 A 类、B 类和 C 类为基本类，D 类用于多播，E 类属于保留类，它们的格式如图 9-2 所示。若要区分一个 IP 地址所属的类别，只需要看它的第一个十进制整数。

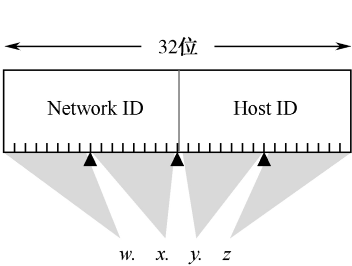


图 9-1　IP 地址格式

0 1 2 3　8　16　24　31

A类	0	网络号	主机号		
B类	1	0	网络号	主机号	
C类	1	1	0	网络号	主机号
D类	1	1	1	0	组播地址
E类	1	1	1	1	备用

图 9-2　IP 地址格式及分类

表 9-1 列出了 A 类、B 类和 C 类地址的起止范围。

表 9-1　各类 IP 地址的范围

地 址 类 型	起 止 范 围
A 类	0.0.0.0～127.255.255.255
B 类	128.0.0.0～191.255.255.255
C 类	192.0.0.0～223.255.255.255

IP 地址的分类设计方式灵活、实用，可以有效避免 IP 地址资源的浪费。例如 A 类地址可表示的网络数目非常少，所以常指定给一些比较大的组织和政府机构；B 类可表示网络数目相对来说比较多，这些地址一般分配给一些比较有规模的组织机构；C 类地址网络总数最多，通常分配给一些中小型规模的组织。

除了上面三种类型的 IP 地址外，还有几种特殊类型的 IP 地址。

1）私有地址

私有地址可以用于单位自己组网，但使用私有地址的计算机要上公网必须转换成合法的 IP 地址，因为 Internet 网没有这些地址的路由。Internet 管理委员会规定如下地址段为私有地址：A 类 10.0.0.0 的一个网络，B 类 172.16.0.0～172.31.0.0 的 16 个网络，C 类 192.168.0.0～192.168.255.0 的 256 个网络（如表 9-2 所示）。

表 9-2　私有地址范围

地 址 类 型	私有地址范围	网 络 个 数
A 类	10.0.0.0	1
B 类	172.16.0.0～172.31.0.0	16
C 类	192.168.0.0～192.168.255.0	256

2）网络号为 127 的地址保留为内部回送地址

A 类网络地址 127 是一个保留地址，用于网络软件测试及本地机进程间通信，称为回送地址，例如，127.0.0.1 表示本机地址。

3）主机号全为 1 的 IP 地址表示一个网络的广播地址

所谓广播，表示将信息发送给该网络上的每个主机。TCP/IP 规定，主机号全为 1 的网络地址用于广播，称为广播地址。

4）主机号全为 0 的地址标识一个网络

TCP/IP 协议规定，主机号全为 0 的 IP 地址不分配给任何主机，仅用于表示某个网络的网络地址。

由以上规定可以看出，这几类特殊的 IP 地址在 TCP/IP 协议中有特殊含义，一般不能用作一台主机的有效网络地址。

3．子网掩码

子网掩码可以用来区分一个 IP 地址的网络号和主机号各占多少位。A、B、C 类 IP 地址的标准子网掩码分别为：

A 类—255.0.0.0

B 类—255.255.0.0

C 类—255.255.255.0

通过用 IP 地址与相应的子网掩码进行“与”运算，可以区分出网络号和主机号包含的位数。网络号和主机位计算示例如下：

设某台主机的 IP 地址为 202.196.240.8，使用默认子网掩码，试分别求出网络号和主机号包含的位数。

这个地址属于 C 类 IP 地址，其子网掩码为 255.255.255.0，通过逻辑与运算后结果为 202.196.240.0，是一个网络地址（运算过程如图 9-3 所示）。

IP地址	11001010	11000100	11110001	00001000
子网掩码	11111111	11111111	11111111	00000000
与运算结果	11001010	11000100	11110001	00000000

202.196.240.0

图 9-3　IP 地址和子网掩码与运算过程和结果

其网络和主机号分别是 24 位和 8 位。若另有一台主机的 IP 地址为 202.196.240.55，子网掩码为 255.255.255.0，则同样会得到网络号为 202.196.240.0，因此这两台主机在同一网段内。通过子网掩码来判断两个 IP 地址是否属于同一网络，这在网络传输中相当重要，因为具有相同网络号的主机可以直接通信，不同网络号的主机则需要通过网关或路由器来转送信息。

9.1.2　端口及服务

1．端口

一个进程和另一个进程进行通信时，需要以某种方式标识自己。对此，TCP/IP 用一个被称作端口（Port）的逻辑结构来实现。

一个端口号是一个 16 位的数字，它唯一地标识一个进程。每个 TCP 或 UDP 分组头都包含源和目的端口号。

当一个数据报到达目的地时，IP 检查该数据使用的协议是 TCP 还是 UDP，上交给传输层协议的数据包含目的端口号，它告诉传输层协议数据应该交给哪一个应用进程。

端口号在/etc/services 文件中描述。0～1023 的端口号被称作约定公用端口，保留为标准的应用程序使用，例如，FTP 使用 21，telnet 使用 23，http 使用 80 等。用户应用程序应使用其他的端口号进行通信。

2．服务

服务是协议和端口的组合。Linux 作为一个网络操作系统最主要的功能就是提供各种网络服务。各种服务被定义在文件/etc/services 中，此文件规定了服务和端口与通信协议的对应关系。

9.1.3　物理地址与逻辑地址

1．物理地址

物理地址（MAC 地址）一般位于网卡中，一个网卡具有全球唯一的 MAC 地址，用于标识网络设备，控制对网络介质的访问。例如，网络设备要访问传输电缆（网线，位于物理层），必须具备一个 MAC 地址，发送的数据要到达目的地，必须知道目的地的 MAC 地址。

2．逻辑地址

逻辑地址（网络地址）即 IP 地址或网络地址。IPv4 使用 32 位二进制来表示网络地址，称为 IP 地址或逻辑地址。逻辑地址用于网络层上对目的主机的寻址。

9.1.4　主机名及设置

尽管通过 IP 地址可以识别主机上的网络接口，进而访问主机。但是 IP 地址不便记忆，对于用户来讲更常用的还是主机名。在 TCP/IP 中，通过域名系统 DNS 来提供主机名和 IP 地址之间的映射信息。

在 UNIX/Linux 系统中，通常使用 hostname 和 hostnamectl 命令来显示或设置系统的主机名。

1．hostname

hostname 功能是显示和临时设置主机名，其语法为：

```
hostname [-v] {name|-F file}            #（从文件 file 中）设置主机名
hostname [-v] [-d|-f|-s|-a|-i|-y|-n]    #按指定格式显示
hostname [-v]                           #显示主机名
```

hostname 的部分参数如表 9-3 所示。

表 9-3　hostname 的部分参数

参　　数	意　　义	参　　数	意　　义
name	主机名	-F file	从文件 file 中读取拟设置主机名
-s,--short	短格式	-f,--fqdn,--long	长格式，全名（FQDN）
-a,--alias	别名	-d,--domain	域名
-i,--ip-address	IP 地址	-y,--yp,--nis	NIS/YP 域名

说明：hostname 设置的主机名没有被保存到/etc/hostname 文件中，在下次设置或系统重启前有效。在系统重启后恢复为/etc/hostname 中设置的值。

2．hostnamectl

hostnamectl 用于查询和设置主机名，其用法为：

```
hostnamectl [options] {cmd}
```

常用的选项和子命令如表 9-4 所示。

表 9-4　hostnamectl 常用的选项和子命令

选　　项	意　　义	命　　令	功　　能
-h/--help	显示帮助信息	status	显示当前主机设置信息（默认）
--transient	修改临时主机名	set-hostname NAME	设置主机名为 NAME
--static	修改静态主机名	set-icon-name NAME	设置图标名为 NAME
--pretty	修改漂亮主机名	set-chassis NAME	设置主机类型名为 NAME
--no-ask-password	特权操作时不需认证	set-deployment NAME	设置 deployment 为 NAME
-H/--host=[USER@]HOST	远程操作	set-location NAME	设置主机位置为 NAME

hostnamectl 区分 3 个不同的主机名：高级的“漂亮（pretty）”主机名（可包括各种特殊字符，如“Lennart's Laptop”）；用于在启动时初始化内核主机名的“静态（static）”主机名（如“lennarts-laptop”）及从网络配置接收到的“暂时（transient）”主机名。如果静态主机名已设置且有效（除 localhost 外），则不会使用暂时主机名。

设置主机名的示例如下：

```
#hostnamectl                                  #显示当前主机设置信息
#hostnamectl set-hostname linux.for.lerning   #同时设置临时和永久主机名
```

9.1.5　网卡命名方案

Linux 传统对以太网卡的命令采用的是 eth#的办法，如 eth0、eth1 等，但在引入 systemd 之后，网卡的命名变了，如 eno16777736、ens33、enp7s0、wlp3s0 等。

1．网卡名字的前 2 个字符

对于网卡的命名，前 2 个字符采用以下方法：

- en：以太网（Ethernet）
- wl：无线局域网（WLAN）
- ww：无线广域网（WWAN）

2．网卡名字的第 3 个及以后字符

根据设备类型不同，网卡名字的第 3 个及以后字符选择如下。

- o<index>：内置网卡时，index 为内置网卡设备编号。
- s<slot>：外置网卡时，slot 为热插拔口编号。
- x<MAC>：网卡的 MAC 地址。
- p<bus>s<slot>：PCI 几何位置。
- p<bus>s<slot>[f<fun>][d<devid>]：网卡 PCI 插口位置。
- p<bus>s<slot>[f<fun>][u<port>][..][c<config>][i<if>]：USB 端口号链。

3．网卡命名方案

Linux 系统在引入 systemd 后，对网卡的命名方案如下：

方案 1　对于内置网卡，结合硬件或者 BIOS 为设备提供索引号命名，如 enoN。如果硬件和 BIOS 信息可用，则用此方案，否则使用方案 2。

方案 2　对于非内置网卡，结合硬件或者 BIOS 提供的 PCI 热插拔插槽索引号命名，如 ensN。如果硬件和 BIOS 信息可用，则使用此方案，否则使用方案 3。

方案 3　结合硬件连接器物理位置的名称，如 enpMsN。如果可用，使用此方案，否则使用方案 5。

方案 4　使用接口 MAC 地址命名，比如 enx78e7d1ea46da。默认情况下不使用此方案，除非用户选择。

方案 5　传统命名方案，如 ethN，其他命名方案失败时，采用此方案。

若要将不按传统方案命名的网卡改为传统的命名方式，可参见 7.2.4 节，让系统采用传统方案。

9.2　TCP/IP 配置

9.2.1　与网络有关的配置文件

在 Linux 系统中，TCP/IP 网络的运行必须使用许多配置文件，了解这些文件的结构和内容对系统管理员来说是必需的，因为通过修改这些文件可达到配置网络的目的。这些文件有的是属于 TCP/IP 的，有的是属于具体操作系统的。

1．通用的 TCP/IP 文件

在本书所介绍的 Linux 系统中，通用的配置文件有/etc/hosts、/etc/services、/etc/resolv.conf、/etc/host.conf/etc/networks 和/etc/hostname 等。

1）/etc/hosts

/etc/hosts 是将 IP 地址和主机名联系起来的简单文本文件。TCP/IP 网络中的每台计算机都必须有唯一的 IP 地址。hosts 文件允许用户将主机名与 IP 地址对应起来，以便访问其中的主机时使用主机名，而不必输入和记忆一长串的点分十进制数字的 IP 地址。/etc/hosts 的结构为：

```
ip_addr hostname alias
```

其中 ip_addr 为 IP 地址，hostname 为主机名或域名，alias 为别名。某个主机的 hosts 文件部分内容如下。

```
127.0.0.1       localhost localhost.localdomain localhost4 localhost4.localdomain4
::1             localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.254.5    linux.gjshao      linux l9
192.168.254.100      solaris.gjshao sun s10
192.168.254.130      freebsd.gjshao bsd bsd7
```

```
192.168.254.10   linux9a.gjshao a9
192.168.254.1    windows.gjshao win xp
192.168.254.20   fedora24.gjshao f24
```

此文件前 2 个有效行分别为 IPv4 和 IPv6 所使用的本地主机名。IP 地址 127.0.0.1 映射到主机 localhost.localdomain，并且分配给它一个别名 localhost。文件的最后 1 行定义了一个名为 fedora24.gjshao 主机，其 IP 地址为 192.168.254.20，别名为 f24。

尽管 DNS 服务器可以取代此文件用于实现名字解析，但由于以下原因该文件仍然存在：

① 文件中的前 2 个有效行要用于本地内部回送网络地址解析。

② 大多数系统都有一个包含本地网络上关键主机名称和地址信息对应的主机表。当 DNS 未运行时，则使用该表进行地址解析。即使本地 DNS 服务器正在运行，在每个系统上也应该有一个 hosts 文件，其中应包含自己可以直达的主机或路由器。

③ 未连接到因特网或其他网络的小型或专用网络不需要 DNS 服务，但这些网络上的主机也需要知道如何定位同一网络上的其他主机，可通过此文件来实现地址解析。

2）/etc/services

/etc/services 是服务和端口与协议对应文件，或者称服务定义文件，其结构为：

```
servicename          port/protocol alias
```

其中 servicename 为服务名，port 为服务使用端口，protocol 为服务使用通信协议，alias 为服务别名。在此文件中每个有效行定义一个服务，用户也可在此系统中定义自己的服务。

以下是某系统/etc/services 文件的部分内容。

```
#The latest IANA port assignments can be gotten from
#http://www.iana.org/assignments/port-numbers
#The Well Known Ports are those from 0 through 1023.
#The Registered Ports are those from 1024 through 49151
#The Dynamic and/or Private Ports are those from 49152 through 65535
##Each line describes one service, and is of the form:
##service-name    port/protocol    [aliases ...]    [#comment]

tcpmux          1/tcp           #TCP port service multiplexer
tcpmux          1/udp           #TCP port service multiplexer
ftp-data        20/tcp
ftp-data        20/udp
#21 is registered to ftp, but also used by fsp
ftp             21/tcp
ftp             21/udp               fsp fspd
ssh             22/tcp               #SSH Remote Login Protocol
ssh             22/udp               #SSH Remote Login Protocol
telnet          23/tcp
telnet          23/udp
```

从该文件注释中可以看出，端口 0～1023 之间为周知端口（Well Known Ports），在 1024～49151 为注册端口，在 49152～65535 间为自动或私有端口。当用户定义自己所使用的端口或服务时，要按文件格式在 49152～65535 间选择未被使用者，并为自己的服务定义一个与众不

同的名字，比如 myserv   56789/tcp。

3）/etc/resolv.conf

/etc/resolv.conf 是域名服务器定义文件，它包含了主机的域名搜索顺序和 DNS 服务器的地址，每一行应包含一个关键字和由白空格分隔的参数。如果这个文件不存在或是空的，那么系统就假设名字服务器在本地的主机上。下面是 resolv.conf 文件内容的一个实例。

```
nameserver 211.67.112.1
nameserver 202.102.224.68
nameserver 202.102.227.68
search localdomain
```

合法的参数及其意义如下所述。

nameserver：设置域名服务器，最多可以有 3 个 nameserver，每个带一个 IP 地址。在查询时按照 nameserver 在本文件中的先后顺序进行，因此它们分别被称为第一、第二和第三 DNS 服务器。在解析时，当使用前一个无法成功解析时，则会查询下一个。

domain：声明主机的域名，在声明的域内可以使用主机名。该选项可以不设置。

search：定义由白空格分隔域名的搜索列表，最多加入 6 个，通常只有 localhost。

以上内容的关键是 nameserver，若 nameserver 设定错误，将不能实现域名解析。

4）/etc/host.conf

/etc/host.conf 的作用是规定如何解析主机，其格式为每行一个的“命令”关键字，这些关键字的意义如下。

order mechanism：指定主机名查询顺序，其参数为用逗号隔开的查找机制，可以是 hosts（/etc/hosts），bind（DNS）和 nis（NIS）。

trim domains：当 DNS 进行地址到主机名的转换时，域名将从主机名中被去除。trim 可以被多个域包含多次；对/etc/hosts 和 NIS 查询方法不起作用。

multi on/off：指定是否对/etc/hosts 文件中指定的主机可以有多个地址。若为 on 则输出所有地址；否则只输出第一个。

reorder on/off：是否允许对地址重新排序。若为 on，所有的查询将被重新排序，则在同一子网中的主机优先，默认值为 off。

某系统的/etc/host.conf 内容如下：

```
#The "order" line is only used by old versions of the C library.
order hosts,bind
multi on
```

5）/etc/networks

/etc/networks 是一个文本文件，用于定义子网。每个文件有效定义一个子网，格式为：

```
subnet    addr    [alias …]
```

subnet 为子网名称；addr 为点分十进制的子网 IP 地址形式，后面表示主机部分的.0 可以省略；alias 为子网别名，也可以不使用。比如：

```
default 0.0.0.0
loopback 127.0.0.0 lo0
```

link-local 169.254.0.0 local
subnet1 192.168.254 net1

networks 只支持 A、B 和 C 类网络，定义在其中的子网可以被 route 和 netstat 等使用。

6）/etc/hostname

/etc/hostname 存放的是系统的静态主机名，系统启动时读取本文件，并以其内容设置主机名。这表明使用 hostnamectl 等工具设置的静态主机名也将存放在其中。

2. 红帽系统

红帽系列系统使用的配置目录主要是/etc/sysconfig 目录，与网络接口配置相关的文件是/etc/sysconfig/network-scripts 中的 ifcfg-<interface>文件，interface 表示具体的网卡，其中的 ifcfg-lo 为内部回送网络接口，其他的如 ifcfg-eth0 或 ifcfg-ens33 等为具体的网卡配置文件，具体取决于是什么网卡及其安装位置（参见 9.1.5 节网卡命名方案），为了描述方便，我们借用 ifcfg-eth0。

1）网卡配置文件 ifcfg-<interface>

ifcfg-<interface>内可包含很多配置信息，常见的配置项目及意义如表 9-5 所示。

表 9-5　ifcfg-<interface>常用配置项目及意义

配　置　项	意　　义
TYPE=Ethernet	该设备是以太网卡类型
DEVICE=eth0	设备名称
NAME=eth0	网络连接的名字
BOOTPROTO=none/dhcp	手工或动态设置 IP：none 为手工设置；dhcp 为自动
ONBOOT=yes/no	系统重启时是否启动该设备：yes 为启用；no 为禁用
USERCTL=yes/no	禁止普通用户启动或删除该设备：yes 为允许；no 为禁止
IPADDR=IP_FORM	IP 地址
NETMASK=IP_FORM	子网掩码
NETWORK=IP_FORM	网络地址
GATEWAY=IP_FORM	网关地址
BROADCAST=IP_FORM	广播地址
PEERDNS=yes/no	是否允许 DHCP 获得的 DNS 覆盖本地的 DNS：yes 为允许；no 为不允许
UUID=uuid	全球唯一标识
NM_CONTROLLED=yes/no	是否可以由 Network Manager 托管：yes 为托管；no 为不托管
DEFROUTE=yes/no	default route，是否把此设备设置为默认路由：yes 为是；no 为否
PREFIX=LEN	子网掩码位数 LEN
DNS1=IP_FORM	第一个 DNS 服务器 IP
DNS2=IP_FORM	第二个 DNS 服务器 IP

2）/etc/init.d/network

/etc/init.d/network 是传统网络服务的管理启动脚本。早期，此脚本用于系统网络启动管理，方法如下：

```
/etc/rc.d/init.d/network start        #启动网络
/etc/rc.d/init.d/network stop         #停止网络
/etc/rc.d/init.d/network restart      #重启网络
```

在现在的版本中，此脚本文件不再直接使用，可以使用命令 ifup 和 ifdown 启动和关闭指定网卡，也可使用 nmcli 管理网络（见后述）。

3．Ubuntu 系统

Ubuntu Linux 系统采用/etc/network/interfaces 配置文件定义每一个网络接口，所有网络接口参数均位于 interfaces 文件中。/etc/init.d/networking 是网络服务的启动脚本。

1）/etc/network/interfaces

interfaces 配置文件定义的网络接口参数如表 9-6 所示。

表 9-6　interfaces 配置文件定义的网络接口参数

配　置　节	说　　明
iface	定义一个逻辑网络接口名，若 mapping 不存在，逻辑接口名等同于物理接口名
mapping	定义网络物理接口名与逻辑接口名的映射关系
auto	采用“-a”运行 ifup 命令时，系统会自动启用相应网络接口（该选项被系统启动脚本使用）
allow-	用于指定应当自动被各种子系统启动的接口。allow-auto 或 allow-hotplug 表示自动启用相应网络接口
source	用于从别的文件包含配置信息，如 source interfaces.d/machine-dependent
source-directory	用于包含目录中的多个文件，如 source-directory interfaces.d

auto 用于启动一个网络接口，而 iface 用于配置一个网络接口。

当配置一个网络接口时，以 iface 开始，后接逻辑接口名的节用于定义逻辑接口，接口名后面跟的是接口使用的地址族名，再后面是接口的配置方式。额外的配置项可写在接下来的行中。可用的配置项依赖于所使用的地址族和配置方式。简单说，iface 开始的行可写为如下形式：

```
iface 网络接口 网络协议 配置方式
```

“网络接口”为逻辑网络接口名，比如 eth0、eth1 等。

“网络协议”为 IPv4 为 inet、IPv6 为 inet6、IPX 为 ipx。

“配置方式”为对于 IPv4 有 dhcp、manual 和 static 等。

- dhcp 意为通过 DHCP 服务获得地址。
- manual 方式用于定义默认没有配置的接口，这些接口可以通过/etc/network/if-*.d 下脚本的 up 或者 down 命令进行配置。
- static 方式用于人工和手动配置以太网卡静态 IPv4 地址，需要设置的选项有：address IP_FORM、netmask IP_FORM、broadcast IP_FORM、metric METRIC、gateway IP_FORM、pointopoint IP_FORM、hwaddress MAC 和 mtu size 等。这里的 IP_FORM 为点分十进制 IP 地址形式：X.X.X.X，下同。

一个设置 eth0 网卡静态 IP 地址的/etc/network/interfaces 文件内容如下：

```
auto lo                              #自动启用回送
iface lo inet loopback               #内部回送网络接口
auto eth0                            #ifup 运行时，系统自动启用 eth0
```

```
iface eth0 inet static                          #eth0 IPv4 ip 地址静态分配
iface eth0 inet6 auto                           #eth0 IPv6 自动分配
address    202.196.233.54                       # IP 地址
netmask    255.255.255.0                        #子网掩码
gateway    202.196.233.254                      #网关地址
```

若需要在一个逻辑网络接口上设置多个 IP 地址（虚拟 IP 地址），可以在 interfaces 文件中添加如下行：

```
auto eth0:1
iface eth0:1 inet static
address X.X.X.X
netmask X.X.X.X
gateway X.X.X.X
```

2）/etc/init.d/networking

/etc/init.d/networking 是传统 Ubuntu 网络服务管理脚本，在早期可以通过该脚本进行网络启动管理，方法如下：

```
$ sudo /etc/init.d/networking start             #启动网络
$ sudo /etc/init.d/networking stop              #停止网络
$ sudo /etc/init.d/networking restart           #重启网络
```

在现在的版本中，此脚本文件不再直接使用，可以使用命令 ifup 和 ifdown 启动和关闭指定网卡，也可使用 nmcli 管理网络（见后述）。

9.2.2 网络服务及管理

NetworkManager 是红帽和 Ubuntu 系统当前版本中的默认网络服务管理器（见 8.1.3 节）。NetworkManager 可以用于 Ethernet，VLANS，Bridges，Bonds，Teams，Wi-Fi，mobile broadband（如移动 3G）以及 IP-over-InfiniBand 等类型的连接管理。针对这些网络类型，NetworkManager 可以配置它们的网络别名、IP 地址、静态路由、DNS、VPN 连接以及很多其他的特殊参数。利用网络管理器程序可以自由切换在线和离线模式。网络管理器会相对无线网络优先选择有线网络，支持 VPN。

1．网络服务管理工具

NetworkManager 由后台服务进程和前台用于不同界面的小程序组成。前者用来管理连接并汇报连接变化，主要实现在系统启动与关闭时对网络设备的探测、启动、关闭、对各种网络设备底层接口的封装等工作；后者允许用户管理网络连接。

用户不能直接操作网络管理系统的服务，需要通过前台界面或命令行工具才能进行网路服务管理，常见工具有 4 种，如表 9-7 所示。

表 9-7　网络服务管理工具

工具及应用程序	描　　述
nmcli	命令行工具（参见 8.1.3 节）

（续表）

工具及应用程序	描　　述
nmtui	一个简单的基于文本用户的 TUI 界面
nm-connection-editor	图形界面管理工具，可在 CLI 或按 ALT+F2 键执行该命令，进入网络编辑界面
control-center 界面	GNOME 环境下用户界面工具，从设置中心选择“Network”进入

2．网络及网络服务管理

网络管理器守护进程运行需要 root 权限，在默认情况下，配置在开机时启动。

1）查看网络服务状态

```
#systemctl status NetworkManager
```

2）启动网络服务

如果 NetworkManager 服务没有处于 Active 状态，以 root 用户启动它。

```
#systemctl start NetworkManager
```

3）启用服务

每次启动系统自动启动网络管理器命令：

```
#systemctl enable NetworkManager
```

4）开启、关闭或重启网络

可用命令行工具 nmcli 开启或关闭网络，nmcli 的用法为

```
nmcli n[etworking] on | off | connectivity [check]
```

使用关闭、开启和重启网络的命令如下：

```
#nmcli n on                                              #开启
#nmcli n off                                             #关闭
#nmcli n off; nmcli n on                                 #重启
#nmcli dev wifi connect <name> password <password>       #连接 WiFi
#nmcli dev disconnect iface eth0                         #断开一个接口:eth0
#nmcli r wifi off                                        #关闭 WiFi
```

9.2.3　IP 地址配置

IP 地址配置既可以通过图形界面，也可以通过直接修改配置文件方式进行。

1．图形界面

通过“设置中心”选择“Network（网络设置）”，可以打开如图 9-4 所示网络配置主界面（此为 Fedora 24 下的界面，红帽的其他系统与 Fedora 相同，Ubuntu 系统参照此处理）。

在图 9-4 选择滑块工具，选择“On”可以打开网络连接；选择“Off”可以关闭网络；依次选择“Off”和“On”可以重启网络。选择左下角的“+”或“−”可添加或删除网络设备。选择右下角的“齿轮”可对选中的网络连接进行设置。

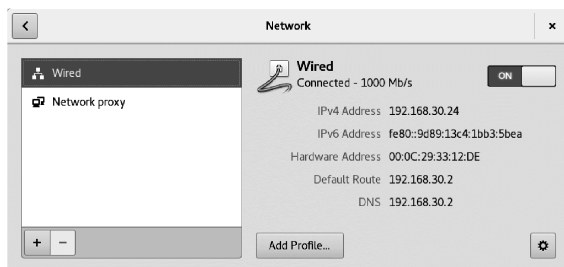


图 9-4　网络配置界面

2．配置现有连接

在图 9-4 中选择现有连接，单击齿轮图标，打开如图 9-5 所示的网络设置界面，从这里可进行网络配置。选择 IPv4，可设为自动获取或手动设置 IP 地址，以及 DNS 和路由配置等。

在“Addresses（地址）”栏的方式下拉菜单中，可选项有“Automatic（DHCP）（自动）”、“Manual（手动）”和“Link-Local Only（仅本地连接）”。若选择“Automatic（DHCP）（自动）”，则只需再选择“Apply（应用）”就可以了，必要时重启一下网络。若选择了“Manual（手动）”，则需要按图 9-5 所示，在“Address（地址）”、“Netmask（掩码）”和“Gateway（网关）”栏填写网络所需的内容，若要上 Internet，还需要在“DNS Server（DNS 服务器）”栏内填上合适的地址。可以在图 9-5 所示界面中，选择“Addresses（地址）”框中右下方的“+”，为同一网卡添加不同的 IP 地址。

还可以在 CLI 界面执行命令 nm-connection-editor，打开如图 9-6 所示的来自 Network-Manager 的网络连接配置界面。选中网络设备后，单击“Edit（编辑）”可打开如图 9-7 所示的网络设置界面。

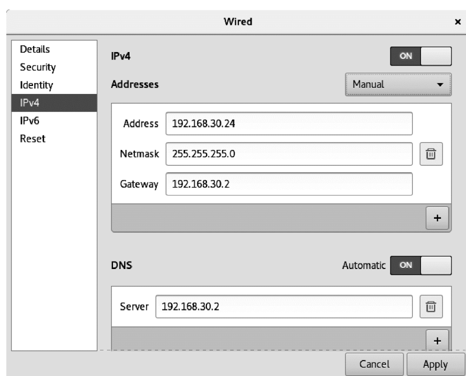


图 9-5　配置现有连接为手动 IP

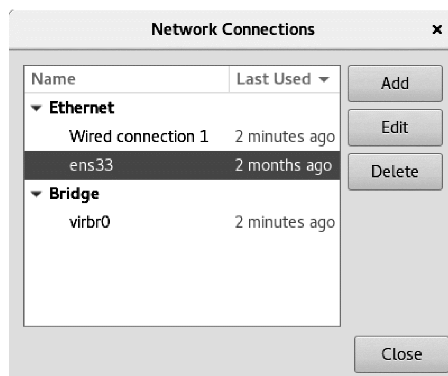


图 9-6　网络连接配置界面

这个界面与 Ubuntu 的网络设置界面完全一样。图 9-7“Method（方法）”的下拉菜单中有更多的可选内容（比如“Automatic（DHCP）”、“Automatic（DHCP）address only”、“Manual”、“Link-Local Only”、“Shared to other computers”和“disabled”），用户可在选择某“Method（方法）”后进行设置，过程同前，不再重复。

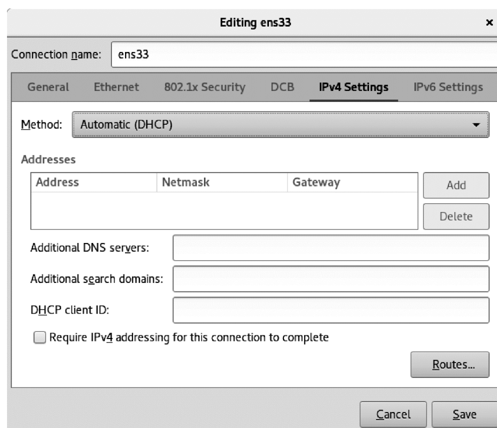


图 9-7　网络设置界面

3．图形界面配置工具与 IP 地址配置文件的关系

1）红帽系统中网络配置中心与配置文件的关系

在红帽系统中，通过图形界面操作的网络信息被直接保存在/etc/sysconfig/network-scripts 目录下 ifcfg-<interface>文件中。每当通过图形界面配置完成后，都将用新配置更新该文件。该文件中与 IPv4 相关部分的主要内容如下：

```
HWADDR=00:0C:29:33:12:DE
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
NAME=eth0
UUID=952c6346-9c2b-31f0-b92c-da4f7fc52c52
ONBOOT=yes
ZONE=FedoraWorkstation
DNS1=192.168.30.2
IPADDR=192.168.30.24
PREFIX=24
GATEWAY=192.168.30.2
```

说明：使用图形界面，尤其 nm-connection-editor，编辑或修改所使用的网络配置文件可能与原来的不同，如 ifcfg-Wired_connection_1 等。

2）Ubuntu 系统中网络配置中心与配置文件的关系

在 Ubuntu 的网络配置工具中配置 eth0 网络，使用静态 IP 地址配置，完成后会将配置信息保存到目录/etc/NetworkManager/system-connections/下的 eth0 文件中。该文件中 IPV4 部分的主要内容如下：

```
[connection]
id=eth0
uuid=0cb2bcf5-df60-435a-b016-65c3cc2977d3
type=ethernet
```

```
timestamp=1493718696
[ipv4]
method=manual
dns=192.168.30.2;
address1=192.168.30.24/24,192.168.30.2
```

由此可见，此文件的格式与/etc/network/interfaces 不同。

需要注意的是，若在配置文件/etc/network/interfaces 中和配置工具中都配置 IP 地址，则 interfaces 和网络管理工具（nm）之间的关系如下：

（1）当系统内没有网络管理工具时，系统默认使用 interfaces 文件内的参数进行网络配置。

（2）当系统内安装了网络管理工具之后，管理工具默认接管系统的网络配置，使用网络管理工具自己的网络配置参数来进行配置。

（3）如果用户在安装网络管理工具之后自己手动修改了 interfaces 文件，那么网络管理工具就自动停止对系统网络的管理，系统改用 interfaces 文件内的参数进行网络配置。此时，再去修改网络管理工具内的参数，不影响系统实际的网络配置。若要让网络管理工具内的配置生效，必须重新启用网络管理工具接管系统的网络配置，操作如下：

① 停止 network-manager 服务

```
$ sudo service network-manager stop
```

② 移除 network-manager 的状态文件

```
$ sudo rm /var/lib/NetworkManager/NetworkManager.state
```

③ 修改文件/etc/NetworkManager/NetworkManager.conf，将其中的行

```
managed=fales
```

修改成

```
managed=true
```

④ 启动 network-manager 服务

```
$ sudo service network-manager start
```

9.3 网络管理命令

9.3.1 测试网络是否通（ping）

ping 命令用来检测和远程主机间的通信连接是否正常，其原理是通过向对方发送一些小的数据包，并等待对方应答来确定两台计算机之间的网络是否连通。如果 ping 不通，则可以预测的故障可能出现在以下几个方面：网线是否连通，网络适配器配置是否正确，IP 地址是否正确等；如果 ping 通了而网络仍无法使用，那么问题可能出在网络系统的软件配置方面。ping 通了只说明当前主机与目的主机间存在一条连通的物理路径。

ping 命令的用法为：

```
ping [options] hostname/ip
```

其参数如表 9-8 所示。

表 9-8　ping 命令常用参数

选　　项	功 能 描 述
-c count	指定发送包的次数为 count。默认为无限次，可用 Ctrl_C 终止
-f	极限检测，每秒钟发送相当于网络处理能力的数据包
-i interval	指定收发信息的间隔秒数
-q	不显示指令执行过程，开头和结尾的相关信息除外
-R	记录路由过程
-s packetsize	设置数据包的大小
-t TTL	将“存活时间”字段设置成数值 TTL 的大小

ping 命令测试网络连接如下：

```
#ping 202.196.233.254 -c 3 -s 1024
PING 202.196.233.254 (202.196.233.254) 1024(1052) bytes of data.
16 bytes from 202.196.233.254: icmp_seq=1 ttl=255 time=0.830 ms
--- 202.196.233.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2182ms
rtt min/avg/max/mdev = 0.830/0.922/1.015/0.097 ms
```

9.3.2　检查网络状态（netstat）

netstat 主要用于显示系统网络信息，包括网络连接、路由表和网络接口通信统计信息等。通过这些信息可以知道开启的端口、正在为哪些用户服务以及服务的状态等，其用法为：

```
netstat [-veenNcCF] [<Af>] -r
netstat [-vnNcaeol] [<Socket> ...]
netstat { [-veenNac] -i | [-cnNe] -M | -s } [delay]
```

其部分参数如表 9-9 所示。

表 9-9　netstat 命令常用参数

选　　项	功 能 描 述
-a	显示所有 socket
-A addr_family	显示地址组信息，addr_family 可选值为 inet，unix 和 ddp 等
-i	显示所有网络接口信息
-n/-N	直接使用数字/符号方式，显示网络连接信息
-r	显示核心路由表，格式同“route -e”
-s	协议通信总量统计
-t	显示 TCP 协议的连接情况
-u	显示 UDP 协议的连接情况

下面介绍 netstat 应用示例。

1．显示处于监听状态的所有端口

如果要显示所有处于监听状态的端口，可使用-a 参数。本示例显示系统内所有 tcp 端口的状态信息。

```
#netstat -ta              #显示所有 TCP 端口
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:32768                 *:*                     LISTEN
… … …
tcp        0      0 127.0.0.1:smtp          *:*                     LISTEN
tcp        0      0 202.196.233.54:34992 222.88.90.209:http      ESTABLISHED
… … …
tcp        0      0 202.196.233.54:35011 58.215.64.145:http      ESTABLISHED
```

输出表明，有多个服务器正在监听（LISTEN），也可以看到目前主机和多个 IP 地址（如 222.88.90.209、58.215.64.145 等）之间已建立起连接（ESTABLISHED）。

2．显示网络接口使用的统计信息

使用–i 选项执行 netstat 可以显示所有已配置接口的一些有用的统计信息，使用该参数，可以很容易检查连接的状态以及连接是否“正常”。因此，这是一个用于检测和排除网络故障的非常有用的措施。

```
#netstat -i
Kernel Interface table
Iface   MTU   Met RX-OK  RX-ERR  RX-DRP RX-OVR TX-OK  TX-ERR TX-DRP TX-OVR  Flg
eth0   1500   0   35014    0       0      0     8810     0      0      0    BMRU
lo     16436  0   82802    0       0      0    82802     0      0      0    LRU
```

MTU（Maximum Transmission Unit）和 Met（Metric）字段表示的是接口的 MTU 和距离值；RX 和 TX 这两列表示的是已经准确无误收发了多少数据包（RX-OK/TX-OK）、产生了多少错误（RX-ERR/TX-ERR）、丢弃了多少包（RX-DRP/TX-DRP），由于错误而丢失了多少包（RX-OVR/TX-OVR）；最后一列显示的是为这个接口设置的状态标记，这些标记一个或几个的字母组合。可用的状态字母及意义如下：

B——已经设置了一个广播地址；

L——该接口是一个回送设备；

M——接收所有数据包（混乱模式）；

N——避免跟踪；

O——在该接口上，禁用 ARP；

P——一个 PPP 链接；

R——接口正在运行；

U——接口处于“活动”状态。

3．显示路由表

```
#netstat -nr        #带参数-r 运行 netstat 将显示内核路由表，其功能和 route -e 命令一样
Kernel IP routing table
Destination     Gateway           Genmask         Flags   MSS Window   irtt Iface
202.196.233.0   0.0.0.0           255.255.255.0   U       0 0          0 eth0
169.254.0.0     0.0.0.0           255.255.0.0     U       0 0          0 eth0
127.0.0.0       0.0.0.0           255.0.0.0       U       0 0          0 lo
0.0.0.0         202.196.233.254 0.0.0.0           UG      0 0          0 eth0
```

在 netstat 输出结果中，第一列为目的地址，第二列是路由条目所指的网关，如果没有使用网关，就会出现一个星号（*）或者 0.0.0.0；第三列为掩码；第四列显示了不同的状态标记；Iface 栏表示该连接所用的物理网卡，如 eth0，eth1，lo 等。

9.3.3　地址解析协议缓存中的项目管理（arp）

arp 用于管理内核缓存的内容。ARP 缓存中包含一个或多个表，它们用于存储 IP 地址及其经过解析的物理地址，其语法为：

```
arp [-v] [-t hwtype] -a [hostname]
arp [-v] [-t hwtype] -s hostname hwaddr
arp [-v] -d hostname [hostname...]
```

这里所有的 hostname 参数可以是符号主机名或是以点分十进制表示的 IP 地址，其参数如表 9-10 所示。

表 9-10　arp 命令常用参数

参 数 名 称	说　　明
-a [hostname]	显示指定主机上所有接口的当前 ARP 缓存表
-e	以默认（Linux）格式显示 ARP 信息
-d hostname	删除与给定主机相关的所有 ARP 信息
-i if	显示指定网络接口的 ARP 信息
-n/-N	以 IP 地址/主机名方式显示信息，默认为-N
-s hostname hwaddr	将 hostname arp 地址加入 ARP 表中。hwaddr 参数指定硬件地址，默认时指的是以太网地址
-t hwtype	限制显示信息为指定的硬件类型，可以是 ether、ax25 等

arp 使用示例：查看本机 ARP 缓存内容

```
#arp
Address       HWtype     HWaddress            Flags Mask        Iface
router1       ether      00:D0:04:CF:3C:0A    C                 eth0
```

9.3.4　网络接口配置（ifconfig）

ifconfig 的功能是管理网络接口，用于查看、配置、启用或禁用网络接口，其用法为：

```
ifconfig [interface]
ifconfig interface [aftype] options | address …
```

ifconfig 参数如表 9-11 所示。在设置中，通常设置的项目有 IP 地址、子网掩码及硬件的中断 IRQ 和 I/O 地址等，其他内容取默认值。

表 9-11　ifconfig 命令的主要参数

字 段 名 称	说　　明
add addr/prefixlen	设置网卡上的 IPv6 地址
[-]arp	停用或应用 ARP（地址解析协议）功能
del addr/prefixlen	删除网卡上的 IPv6 地址
down	关闭指定的网络接口
hw class address	设置指定接口设备的 MAC 地址，目前支持的硬件类型有 ether，ax25 等
io_addr addr	设置网卡的 I/O 地址
irq addr	设置指定接口设备使用的中断行
media type	设置网络设备所使用的介质
mem_start addr	设置网卡在主存储器所占用的起始地址
metric N	设置数据包转送的次数
mtu N	设置网卡的最大传输单位
netmask addr	设置接口的子网掩码
interface	网络设备名，如 eth0 就表示本机的第一块网卡
tunnel aa.bb.cc.dd	IPv4 和 IPv6 间的隧道通信地址
up	激活指定的网络接口卡
[-]allmulti	是否接收 multicast 的数据包，加负号-表示接收
[-]broadcast [addr]	将送往指定地址的数据包视为广播数据包
[-]pointopoint [addr]	启用点对点方式，和指定的地址建立直接连接
[-]promisc	是否应用网卡上的 promiscuous，加负号表示不应用

ifconfig 使用示例如下：

1．显示网卡的配置信息

```
#ifconfig                #不带参数运行 ifconfig 命令可以显示所有网卡的配置信息
#ifconfig eth0           #显示指定网络接口 eth0 信息
#ifconfig eth0:1         #显示指定网络接口 eth0:1 信息
```

2．启动或停用网卡

若要停用某网卡，可以指定网卡名称，并使用“down”参数，要重新启动网卡，则需使

用“up”参数。

```
#ifconfig eth0 down            #停用 eth0 网卡
#ifconfig eth0 up              #应用 eth0 网卡
```

3．设置网卡配置

（1）修改网卡设置。

```
##把 eth0 网卡的 IP 地址和子网掩码分别改变为 202.196.233.5 和 255.255.255.0。
#ifconfig eth0 202.196.233.5 netmask 255.255.255.0
```

（2）在指定网卡上增加一个 IP 地址。

```
##在指定网卡 eth0 上增加一个地址 202.196.235.0 网络掩码为 255.255.255.0
#ifconfig eth0:1 202.196.235.0 up
##在指定网卡 eth0 上增加一个网络为地址 202.196.238.0，网络掩码长度为 26 位的网络
#ifconfig eth0:2 202.196.238.0/26 up
```

9.3.5 网络 IP 管理（ip）

ip 是一个强大的网络配置工具，它能够代替一些传统的网络管理工具，如 ifconfig、route 等，使用权限为超级用户。ip 来自软件包 iproute（红帽）或 iproute2（Ubuntu），其用法为：

```
ip [options] object {cmd | help}
```

ip 的常用参数如表 9-12 所示。

表 9-12 ip 命令的常用参数

类 型	项 目	意 义	项 目	意 义
options	-V | -Version	显示版本	-h | -human-readable	易读方式
	-s | -statistics	输出更多的信息，可多次使用	-d | -details	输出详细信息
	-f <FAMILY>	指定协议族，如 inet、inet6 等	-r | -resolve	使用主机名而非 IP
	-4	-f inet	-6	-f inet6
	-a | -all	在所有 object 上执行 cmd	-c | -color	使用彩色输出
object	link	网络设备	address	IP 地址
	route	路由	tunnel	隧道
cmd	应用于 object 上操作，因对象不同而异			

1．接口设备相关设置

ip link 命令可以设置与设备有关的参数，包括启动或关闭某个网络接口，语法如下：

```
#ip [-s] link show                             #查看设备相关信息
#ip link set [device] [up|down|name|address]   #按照指定动作设置设备
```

ip link 命令使用示例如下：

```
#ip -s link show eth0                          #显示本机 eth0 接口的信息
#ip link set eth0 up                           #启动本机 eth0 接口设备
#ip link set eth0 name v0                      #修改 eth0 网卡名称为 v0
#ip link set eth0 address 52:54:00:fb:77:9e    #修改 MAC 地址
```

2．为接口分配 IP 地址

ip addr/address 命令可用于将 IP 地址分配给接口，语法如下：

```
#ip address show                               #查看 IP 参数
#ip address [add|del] address dev ifname       #设置与 IP 相关各项参数
```

ip addr 命令使用示例如下：

```
#ip address show eth0                          #显示本机 eth0 的 IP 参数
#ip address add 192.168.1.200/24 dev eth1      #添加虚拟接口并分配 IP
#ip address del 192.168.1.200/24 dev eth1      #删除刚刚添加的接口
```

3．设置路由

如果需要静态路由，可以通过 ip route add 命令将路由添加到路由表中，并使用 ip route del 命令移除。语法如下：

```
#ip route show                                                      #查看当前路由设置信息
#ip route [add|del|change|append|replace] destination-address      #设置路由
```

ip route 命令使用示例如下：

```
#ip route add 192.168.10.0/24 dev eth0                         #添加路由
#ip route add 192.168.8.0/24 via 192.168.10.100 dev eth0       #增加可以通往外部的路由
#ip route add default via 192.168.1.254 dev eth0               #添加默认路由
#ip route del 192.168.10.0/24                                  #删除路由
```

9.3.6 网络接口的启用与停止（ifup/ifdown）

ifup、ifdown 与 ifconfig 一起位于目录/sbin 内，是两个用于网络接口管理的脚本程序，在执行过程中都要执行位于/etc/sysconfig 目录中的相关脚本程序，它们的直接作用是启用和停用网卡。比如：

```
#ifup eth1                  #启用网卡 eth1
#ifdown eth1                #禁用网卡 eth1
```

9.3.7 路由表维护（route）

route 命令主要功能是管理 Linux 系统内核中的路由表。它最大的用途就是用来设定静态的路由表项，通常是在系统用 ifconfig 配置网络接口（如网卡等）后，用它来增加、删除或修改路由等，其用法为：

```
route [-add][-net|-host] targetaddress [-netmask Nm]
route [-del][-net|-host] targetaddress [gw Gw] [-netmask Nm]
```

route 命令参数如表 9-13 所示。

表 9-13　route 命令主要参数及说明

参 数 名 称	说　明
-n	使用数字地址形式代替解释主机名形式来显示地址
-e	将产生包括选路表所有参数在内的大量信息
add	添加一条路由
del	删除一条路由
dev	强制路由与指定的设备关联，否则内核自己会试图检测相应的设备
target	配置目的网段或者主机
-net	路由到达的是一个网络
-host	路由到达的是一台主机
netmask	为添加的路由指定网络掩码
gw	指定路由的网关

route 命令使用示例如下。

1．监视路由表内容

```
#route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
202.196.233.0   *               255.255.255.0   U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
```

前两项是通过接口 eth0 到网络 202.196.233.0 和网络 169.254.0.0 的路由。第三项是到 localhost 的回送路由。它们都设置了 U（启动）标志，这表示准备使用它们，但它们都未设置 G（网关）标志，这是因为这两个路由都是通过本地接口。其他标志字段的说明如表 9-14 所示。

表 9-14　路由信息中包含的字段说明

字 段 名 称	说　明
Destination	目标网段或者主机，“default”指本机网关
Gateway	网关地址，如果没有设置，则用"*"表示
Genmask	网络掩码
Flags	U（Up）：路由是活动的；H（Host）：目标是一个主机；G（Gateway）：使用网关；R（Reinstate Route）：动态路由产生的表项；D（Dynamically）：路由动态写入；M（Modified）：路由动态修改；!：被拒绝的路由
Metric	路由距离，表示源主机将信息送达目的主机所需的转达次数
Ref	路由项引用次数

（续表）

字段名称	说　明
Use	查找路由项的次数
Iface	该路由项对应的网络接口
MSS	默认的 TCP 最大分片尺寸
Window	默认的 TCP 窗口的尺寸
irtt	默认的 TCP 回路时间

2．添加路由记录

通过 eth0 增加一条到网络 218.196.240.0/24 的网络。

```
#route add -net 218.196.240.0 netmask 255.255.255.0 dev eth0
#route                                    #查看路由表信息
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
202.196.233.0   *               255.255.255.0   U     0      0        0 eth0
218.196.240.0   *               255.255.255.0   U     0      0        0 eth0
169.254.0.0     *               255.255.0.0     U     0      0        0 eth0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         202.196.233.254 0.0.0.0         UG    0      0        0 eth0
```

3．删除路由记录

删除刚增加的路由信息。若要从路由表中删除某个路由记录，需要使用“del”参数。

```
#route del -net 218.196.240.0 netmask 255.255.255.0 dev eth0
```

9.3.8　主机及地址查询（nslookup/host）

nslookup 和 host 命令的功能是查询一台网络主机的 IP 地址或其对应的域名，通常需要一台域名服务器来提供服务，其用法为：

```
nslookup [域名/IP 地址]
host [域名/IP 地址]
```

host 的应用示例如下：

```
#host hncj.edu.cn                    #正向解析
#host -a www.163.com
#host 222.138.255.18                 #逆向解析
```

nslookup/host 使用示例如下：

```
#nslookup www.hncj.edu.cn            #直接查询：正向
#nslookup 202.196.233.254            #直接查询：逆向
#nslookup                            #进入 nslookup 交互界面
```

##在符号“>”后面输入要查询的 IP 地址或域名并回车即可。如果要退出，输入 exit 并回车。

```
Note:    nslookup is deprecated and may be removed from future releases.
Consider using the `dig' or `host' programs instead.    Run nslookup with
the `-sil[ent]' option to prevent this message from appearing.
> 202.196.233.254                   #逆向解析 202.196.233.254
> www.hncj.edu.cn                   #正向解析 www.hncj.edu.cn
>server 211.67.112.1                #指定 DNS 服务器
>mail.163.com
>exit                               #退出
```

9.3.9 网络路由跟踪（traceroute）

traceroute 是一个通用的 TCP/IP 工具，它能显示数据包从本地机到达目标机时经过的所有路由器。由于 traceroute 会人为地给网络造成一些通信负担，所有它主要被系统或网络管理员作为一个简单而有效的网络手工查错工具使用，普通用户最好少用。尽管如此，traceroute 对于用户了解 Internet 的结构，了解自身的网络故障会有很大帮助，其基本用法为：

```
traceroute [域名/IP 地址]
```

9.4 网络应用常用命令简介

9.4.1 远程登录

telnet 和 ssh 是 UNIX/Linux 系统用于远程登录与访问的客户端工具软件。它们的使用都需要相应网络服务的支持，对应的网络服务分别为 telnet 和 sshd。

1．telnet

用户使用 telnet 命令可以进行远程登录，当用户通过网络成功在远程系统上登录后，就像在本地系统上一样使用系统。

telnet 的客户端程序为 telnet，软件包名也是 telnet；telnet 的服务器端软件包在红帽系统和 Ubuntu 系统下分别为 telnet-server 和 telnetd。如果系统还没有安装，可执行以下命令安装它们：

```
#yum install telnet telnet-server           #红帽系统
#apt-get install telnet telnetd             #Ubuntu 系统
```

关于非独立 telnet 服务器的配置可参见 6.4.3 节，但要开启 xinetd 服务。

在使用 telnet 命令前首先查看其运行状态：

```
#netstat -a | grep telnet
tcp        0      0 *:telnet        *:*        LISTEN
```

telnet 正常运行后可以使用 telnet 远程登录，命令的一般形式：

telnet [主机名/IP 地址]

其中主机名或 IP 地址分别是要连接的远程机的主机名或 IP 地址。为了登录到远程计算机上，必须知道远程机上的合法用户名和口令。使用完毕后可按本地系统注销的办法退出系统。当不带参数运行 telnet 时进入交互界面，在交互界面下，可以通过 help 或?得到帮助。

telnet 是不安全的，因为它的传输过程是非加密的，现在较流行且安全的远程登录系统为 ssh。

2．ssh

ssh 也像 telnet 一样允许用户在远程系统上登录并按权限使用系统。它使用加密的方式来传送所有数据，是 rlogin 和 telnet 等的安全替代品。该命令的一般应用形式为：

ssh 主机名/IP 地址

关于对 ssh 的更进一步讨论可参见本书第 14 章。

9.4.2　文件传输

ftp 和 tftp 是网络上用于文件传输的基本工具，它们工作需要相应服务程序的支持。关于 ftp 和 tftp 的详细介绍参见第 13 章。

1．ftp

ftp 的用途是文件传输，使用前提是用户必须知道远程计算机上的合法用户名和口令，ftp 也提供匿名登录，其用法为：

ftp [选项][主机名/IP 地址]

用户可以通过使用 ftp 客户程序连接到另一台计算机上，登录成功后可以列目录内容、改变工作目录，可以进行文件的下载或上传等。当不带参数运行 ftp 时进入交互界面。在交互界面下，可通过 help 命令得到简单的帮助，还可通过 help cmd 获得对命令 cmd 的帮助。关于 ftp 服务器建设和 ftp 命令的使用请参见本书第 17 章。

2．tftp

tftp 是简单文件传输文件工具程序，它不需要密码和用户验证，因此它的使用受到很多限制。关于 tftp 服务器建设和 tftp 命令的使用请参见本书第 17 章。

9.4.3　邮件收发

mail/mailx 是 Linux/UNIX 系统中比较简单的快速电子邮件用户端软件，它需要 sendmail 服务器的支持，其用法为：

```
mail [-s subject] [-c address] [-b address]
mail -f [mailbox]
mail [-u user]
```

其部分参数如表 9-15 所示。

表 9-15　mail 命令的部分参数

选　　项	功 能 描 述
-b address	指定（由逗号分隔）的收信人地址清单
-c address	指定（由逗号分隔）的抄送人地址清单
-f [mailbox]	从指定邮件文件（默认为 mbox）中读取邮件
-s subject	指定邮件标题
-u user	从用户默认邮箱读取邮件，等同于-f /var/spool/mail/user

当不带参数运行 mail 时，进入 mail 的命令交互界面，mail 的提示符为&。在命令界面可用的 mail 命令如表 9-16 所示。

表 9-16　mail 的内部命令

命　　令	意　　义
?|help [ftp_cmd]	帮助。显示 mail 操作命令
exit|quit|ctrl_D	退出 mail
h	重新显示邮件头部信息列表（headers）
数字或$	按数字选中邮件，$表示最后一个
+/-	下/上一个邮件
p/t<#1 #2 …>	显示第#1,#2,…邮件（以 more 格式阅读）
s<#1 #2 …> <file>	将第#1，第#2，…邮件的内容追加到文件 file
d<#1 #2 …>	删除邮件#1，#2，…
u<#1 #2 …>	被删除的邮件
m<MailBox>	向邮箱 MailBox 发送邮件
R<#>	回复第#号邮件
q	将读过的邮件移到 mbox，删除已标有删除标志的邮件，然后退出
x	直接退出 mail 系统，而不向 q 做更多的事情
sh	启动 sh，可按 Ctrl_D 键返回 mail
!cmd	执行 shell 命令 cmd

mail 的使用示例如下：

例 1　启动 mail 程序并列出邮箱中的消息，可以使用 mail 命令对邮件进行操作。

```
#启动后 mail 列出在系统邮箱中的每条消息。显示“&”提示符表示等待命令输入。
#可以输入“?”获得简单帮助。
#mail
Mail version 8.1 6/6/93.   Type ? for help.
"/var/spool/mail/root": 4 messages 2 new 4 unread
 U   1 root@localhost.local   Thu Jan 18 04:02    74/2169    "LogWatch for localhos"
 U   2 root@localhost          Fri Jan 19 20:38 106/3040    "LogWatch for localhos"
>N   3 root@localhost          Sat Jan 20 12:31    88/2394    "LogWatch for localhos"
```

```
N    4 root@localhost              Sun Jan 21 04:03 1254/48616 "LogWatch for localho"
&
```

例 2　将消息 letter 发送到收件人 user1@host1 并抄送给 user2@host2。

```
#mail -c user2@host2 user1@host1 < letter
```

9.4.4　文件下载

wget 是一个 Linux 环境下用于从 Internet 上提取文件的工具，wget 支持 HTTP 和 FTP 协议，也支持代理服务器和断点续传功能，并能够自动递归远程主机的目录，找到合乎条件的文件并将其下载到本地。由于 wget 是非交互式的，故可在后台运行，捕获并忽略 HANGUP 信号，因此在用户退出登录以后仍可继续运行。wget 可用于成批量下载 Internet 网站上的文件。wget 支持递归操作，并可跟踪 HTML 网页的链接，因此可用于下载完整的目录结构。wget 的用法为：

```
wget [options] [URLs]
```

其部分参数如表 9-17 所示。

表 9-17　wget 的部分参数

命　　令	意　　义
-b	让 wget 在后台运行，记录文件写在当前目录下的"wget-log"文件中
-o/-a log_file	指定 log 文件，并分别以覆盖/追加方式使用
-q	关闭工作过程信息的输出
-i file	指定从文件 file 中读取 URLs 信息
-t num	设置重试次数，默认或为 0 时，表示无限次重试直到成功
-c	断点续传。会断续上次未传输完的任务，但前提是服务器方必须支持此功能
-t secs	指定超时时间，若在 secs 秒内远程服务器没有响应就中断连接，开始下一次尝试
-w secs	指定两次尝试之间等待的秒数
-r	使用递归，用于下载目录
-m	打开站点镜像下载选项。如果想做一个站点的镜像，就使用这个选项

使用 wget 下载数据的示例如下：

```
#下载单个文件
#wget https://github.com/mirror/wget/archive/master.zip
#断点续传
#wget -c https://github.com/mirror/wget/archive/master.zip
```

习题 9

1．思考题

（1）什么是子网掩码？其作用是什么？

（2）何为网络服务？举例说明你所知道的 Linux 系统下的网络服务。
（3）何为 IP 地址？如何查询本机的 IP 地址？如何查询一个互联网上主机的 IP 地址？

2．单项选择题

（1）202.196.100.1 是何类地址（　　）？
A．A 类　　B．B 类　　C．C 类　　D．D 类
（2）当 IP 地址的主机地址全为 1 时，表示（　　）。
A．专用 IP 地址　　B．对于该网络的广播地址
C．本网络地址　　D．回送地址
（3）telnet 和 ssh 使用的端口分别是（　　）和（　　）。
A．20　　B．21　　C．22　　D．23
（4）FTP 工作中所使用的两个端口是（　　）和（　　）。
A．20　　B．21　　C．22　　D．23

3．综合题

（1）与 TCP/IP 相关的文件有哪些？各自的用途是什么？
（2）试述如何管理主机名。
（3）试述如何进行 TCP/IP 网络配置。

实验 9

1．TCP/IP 网络配置。
2．网络管理命令。
3．网络应用常用命令。
4．配置和使用 telnet 服务器。

第 10 章　Linux 系统的安全

10.1　Linux 系统安全概述

Linux 操作系统的安全技术涉及的方面很广，如 BIOS 设定、系统账号及口令安全、重要文件设定、服务端口、各种服务器的安全配置以及日志安全和防火墙设置等。针对于这些安全技术，常见的安全管理主要分为四个方面：防止未授权存取、防止泄密、防止用户拒绝系统的管理和防止丢失信息。

Linux 系统作为一种网络操作系统的应用越来越广泛，因此操作系统的安全也越来越重要。针对于网络环境，操作系统安全主要考虑两个问题：一是计算机病毒，二是黑客对系统的破坏和入侵。

计算机病毒的主要传播途径已经由过去的软盘和光盘等存储介质逐渐变成了网络。这些病毒一旦感染计算机系统，将起到各种各样的破坏作用，给系统带来安全问题。而对于计算机黑客，他们的主要目的在于窃取数据和非法修改系统。其手段之一是窃取合法用户的口令，在合法身份的掩护下进行非法的操作，二是利用网络操作系统中某些合法但不为系统管理员和合法用户所熟知的操作命令。

为了从根本上更好地进行安全防范，需要了解 Linux 系统安全的相关机制、常见安全问题及安全防范策略。

10.1.1　Linux 操作系统的基本安全机制

Linux 系统提供了 BIOS 安全设置、GRUB 安全设置、用户账号认证、系统日志文件和文件系统权限等基本安全机制，这些安全机制都需要系统管理员小心地进行设置。如果系统管理员缺乏安全意识，就会使系统存在一定的安全隐患。

1．BIOS 安全设置

BIOS 是 Basic Input/Output System 的缩写，意为基本输入/输出系统，它提供了底层的硬件支持，早期的 x86 操作系统就是使用它们来访问磁盘、显示器和其他硬件的，但是 UNIX/Linux 系统在获得控制权后并不使用它。从某种意义上讲它对 UNIX/Linux 并没有用处，但是在计算机加电启动的早期，操作系统还没有获得控制之前，并不能直接跳过它。由于在 BIOS 的设置中可以设置密码，因此可以把它作为开机的第一道防线，因为如果不输入正确的 BIOS 密码，计算机就不能继续启动，因而也就进不了操作系统。另外，给 BIOS 设置密码，可以阻止别人进入 BIOS 改动其中的设置，因此可以防止在 BIOS 中改变启动顺序，从而阻止别有用心的人企图用特殊的启动盘启动系统或通过改变启动参数来启动系统，从而有力地保证了计算机软/硬件系统的安全。但这里必须指出，BIOS 设有密码的系统是不能进行远程启动的，在远程维护系统时应该加以注意。

2．GRUB 安全设置

GRUB 是 Linux 的启动模块，可以通过修改其配置文件/etc/grub.conf 中的内容来进行配置。可以在其中使用 passwd 或 lock 命令设置密码，以保护自己的系统。仍需要提醒的是，设置有 GRUB 密码的系统是不能进行远程启动的。

3．Linux 系统的用户账号

目前网络上大部分对系统的攻击都是从截获口令或者猜测口令等口令攻击开始的，所以首先应该对账号和口令的安全进行设置。在 Linux 系统中常见的账号安全设置方法主要有以下几种。

（1）设置默认口令和账号的长度及有效期。

Linux 系统默认的用户密码是 5 位，口令的有效期是 7 天，可以编辑/etc/login.defs 文件，更改口令的长度和有效期。

（2）清除不设口令的账号。

禁止不设口令的账号存在，可以通过/etc/passwd 文件来查看并清除。

（3）特别的账号处理。

禁止所有默认被操作系统本身启动且又不需要的账号。

4．Linux 系统的日志文件

日志是了解 Linux 系统运行情况的唯一方法。系统管理员要合理利用 Linux 系统的日志文件，监控好系统的运行。常用的日志文件有以下三个。

（1）/var/log/lastlog 文件：该文件记录最后进入 Linux 系统的用户信息，有登录时间、是否登录成功等信息。用户再次登录系统后可以使用 lastlog 命令查看 lastlog 文件，确定是否与实际用机情况一致，从而发现该用户账号是否被盗用过。

（2）/var/log/secure 文件：该文件为系统管理员提供系统开通后所有用户的登录时间和登录地点。

（3）/var/log/wtmp 文件：该文件也是记录用户的登录信息，包括当前和历史上登录到系统的用户的登录时间、登录地点和注销时间等信息。可以使用 last 命令查看。

5．Linux 系统的文件系统权限

在 Linux 系统中，每一个文件或目录都有 3 个或 3 组权限属性，分别定义文件或目录的所有者、同组用户和剩余所有其他用户的使用权限。这些权限包括读、写、执行、SUID、SGID 等。Linux 文件系统的安全主要是通过设置文件或目录的这些权限来实现的。当为一个用户分配权限时，应以最小或极小权限为原则，不能过大，还要防止任何人越权行事。尤其是 SUID 和 SGID 属性，允许程序以其他用户身份或超级用户身份运行，会使使用者摇身一变为其他人，因此具有 SUID 和 SGID 的程序不宜过多，若系统中不明原因地增多了此类程序，则说明可能出现了安全隐患或已经造成了危害。

10.1.2　Linux 系统可能遇到的安全问题及防范策略

根据前面提到的 Linux 系统的安全涉及的各个方面，在实际使用过程中都会遇到相关的

问题，例如，系统启动参数被修改、账号与口令被修改、系统被其他用户非正常登录等。但是，Linux 在网络系统中的应用主要与网络攻击相关，下面重点讨论网络安全情况。

1．Linux 系统可能遇到的安全情况

1）黑客攻击

黑客（hacker）是指那些喜欢用计算机干有趣而不寻常事的人，尤其是对计算机程序的源代码进行处理，也就是说，黑客们感兴趣的是源代码或对新事物的学习。大部分黑客并不是想获得对别的系统的非授权访问，而只有一小部分黑客侵入计算机系统干一些非授权活动，以进行非法的、未经许可的访问。因此黑客的范围被扩大了，为了区分，人们把那些干坏事的黑客称为骇客（cracker），这才是真正的黑客。

2）电子欺骗

在网络中还有一些组织或个人采取欺骗的方法装扮为其他个人或伪装成网络公司，使用别人的访问权以谋取不正当的利益，例如，在网络上伪装后向其他用户发出呼叫，并在适当的时候要求用户输入口令以窃取密码或有用信息，或用网络上的主机冒充他人主机的身份，将用户信息送往错误的机器。

3）拒绝服务攻击

当怀有恶意的用户发送大量信息到某计算机上，使授权用户无法执行正常的任务时，就会发生拒绝服务（Denial of Service，DoS）攻击。这种方法主要是恶意用户采取具有破坏性的方法阻塞目标网络的资源，使网络瘫痪。最通常的 DoS 攻击一般是同时发送很多个网络包到某个服务器，掠夺服务器的资源，最后导致合法的用户请求无法通过，使系统无法正常工作。

4）病毒

病毒是一种程序或者是程序的一部分，在不知道或不允许的情况下进行自身复制或通过存储介质及网络传播，它会对计算机或网络造成各种各样的危害。例如，病毒会掠夺所有的物理或虚拟内存，使计算机被挂起或者瘫痪。更厉害的病毒会删除或篡改数据文件或使系统崩溃。病毒的种类很多，防不胜防，黑客们会利用它们很容易地破坏目标系统或从中获取重要信息。

5）扫描程序攻击

黑客或恶意用户会利用扫描工具查找目标计算机上各种各样的漏洞，并用之对系统实施攻击，从而侵入目标系统，要对此类攻击提高警惕，加强防范。

2．设计安全策略

对系统的攻击可能来自非授权用户，但也有可能来自合法用户。因此，Linux 系统管理员既要警惕来自外部的攻击，又要加强对内部用户的管理，另外也要注意硬件安全和误操作。为此，系统管理员可以采用下面的安全防范策略。

1）物理环境保障

物理环境包括防水、火、盗、电磁辐射，物理网络隔离，用电环境，楼房建筑与机房位置等诸多方面。

2）制度保障

建立各种规章制度，并强制执行，定期进行安全检查。

（1）网络隔离制度。一般单位都要求内、外网隔离，业务网与办公网隔离。特殊情况需

要相互渗透时要通过防火墙和中间件。

（2）工作环境管理制度。要有防水、防火、防盗、防电磁辐射，防自然灾害等制度和应急响应机制，要有安全可靠的电力保障。

（3）人员管理制度。要建立外部人员来访管理制度，出入登记制度，系统使用登记制度，岗位分工与管理制度，定期培训与换岗制度。

3）用户密码的安全性

密码是保障系统安全的关键。密码是防止别有用心的人本地和异地登录系统的第一道屏障，为此，一定要保护好密码文件/etc/passwd 和/etc/shadow 的安全，不让无关的人员获得这两个文件。同时，系统管理员还应该告诉用户在设置密码时要使用安全密码（使用非数字或字母的特殊字符）并适当增加密码的长度（大于 6 个字符）。系统管理员还可以定期使用 Crack、John 等破解程序对系统的/etc/passwd 文件和/etc/shadow 文件进行模拟字典攻击，若发现不安全用户密码，要求用户必须修改。

关于密码的管理，还应有一套完整的制度，不论是系统密码还是用户密码都要定期或不定期修改。

4）设置好内部用户的权限

在创建新用户时，要仔细设置每个内部用户的权限和用户所在的组，授权时一般应遵循“最小权限”原则，即仅给每个用户授予完成特定任务所必需的访问权限。

5）检查文件系统的安全性

文件系统的安全性是指限制文件和目录权限，慎重地使用 SUID 和 SGID 权限，检查并删除多余和可疑的文件，并设置系统的默认权限方案。

6）加强对系统运行的监控和记录

系统管理员应该对整个系统的运行状况进行监控和记录。这样，通过分析记录的数据，可以发现可疑的活动，从而及时采取措施阻止可能发生的入侵行为。若系统已经被入侵，利用记录的运行状况数据，可以跟踪和识别入侵的非法用户及途径。

系统中的各种启动日志、管理日志、应用程序日志等也要记录完备。

7）数据备份

因为有人、自然和制度等多方面的因素，至今没有一种操作系统的运行是绝对安全可靠的，也没有一种安全策略是万无一失的，所以要制订合适的数据备份计划和制度，并按制度认真执行之，确保备份数据的有效性和可用性，一旦系统遭到破坏或攻击而发生瘫痪时，能利用备份的数据迅速恢复工作。

8）提高系统的运行级别

很多操作系统可以运行在不同安全级别，提高系统的安全级别，可以增强系统的安全性。不过安全与方便有时是矛盾的，必须在两者之间权衡，但安全是第一位的。

9）开启防火墙

在网络中，“防火墙”是指一种将内网和公网（如 Internet）分开的方法，实际上是一种隔离技术。能通过防火墙规则设置最大限度地阻止网络中的非法或未经授权的访问，提高网络和系统的安全性。强烈建议，在网络上运行的系统要开启防火墙，并合理设置防火墙的规则。

10）配置 SELinux

使用 SELinux（Security Enhanced Linux）进行文件系统的权限设置，从而有效地管理用户存取数据时的行为。

10.2　检查和监督系统的运行情况

10.2.1　检查网络

为了保证系统和网络的正常安全运行，对系统的检查和监督是必需的。ifconfig 命令的功能是对用户的网络接口进行检查和配置。显示系统内所有网络设备使用情况可不带参数运行 ifconfig 命令。

netstat 命令可以用来查看诸如网络连接状态、接口配置信息、路由表、通信统计值等信息。

ifconfig 命令和 netstat 命令具体用法参见第 9 章中网路管理命令部分。

10.2.2　用 ps 或 pstree 检查进程

在 UNIX/Linux 操作系统中可以使用 ps 或 pstree 命令查看进程状态。通过查看进程的状态，可以发现系统中有无多余的进程。如果有比预期多余的进程，可以通过命令 kill 或 killall 将其停止，并进一步从系统中将对应程序删除。

10.2.3　检查系统的日志文件

在 Linux 中，系统的日志文件用于诊断安装、引导、特定服务等问题。这些日志文件包含了系统的运行信息，如内核、服务及系统运行的程序和用户登录情况等。

例如，通过对服务器日志的检查，可以了解系统各种服务的运行情况，以便对服务器进行配置和参数调整，确保系统的正常运行。同时，若发现多余的服务，可以将其停止，以消除系统的安全隐患。

如果发现了多余的安装软件包，应该及时将其删除掉，因为这完全有可能是某个别有用心的人在系统内安装的后门或定时炸弹，说不定在随后的某个时刻就会爆炸。

也可以通过查看系统的登录日志发现是否有用户非法进入系统或试图进入系统。

10.2.4　停止不需要的服务

Linux 系统在第一次安装时总是会在系统中安装一些系统认为可能必要而我们并不需要的服务。其中的有些服务，比如 telnet、ftp 和 Web 服务器等可能是需要的，但还有一些服务，比如 echo、Chargen 和 Daytime 等，通常是不使用，为了系统的安全应当关闭它们。

在实际应用时，可以先使用 netstat、lsof 或 chkconfig --list 对服务进行扫描，并根据检查的结果确定需要关闭哪些不必要的服务。一旦确定系统中不需要特定服务，只需确保该服务不会在系统启动时被运行即可，当然更安全的办法是删除为这些服务提供支持的软件包。

10.2.5 去掉多余的具有 SUID 和 SGID 属性的文件

系统中 SUID 和 SGID 属性的文件将会给执行它们的用户提供一些特权，对于其中不是系统需要的部分要尽可能早地的删除或去掉它们的 SUID 和 SGID 属性。

可以首先使用 find 命令：

```
#find / -perm /6000 -print
```

在系统中查找所有具有 SUID 和 SGID 属性的文件，然后删除不需要的部分，或使用权限管理命令 chmod 去除它们的 SUID 或 SGID 权限。

10.3 入侵检测和事件报告

10.3.1 使用完整性检查工具

入侵检测指的是在系统已经可能被攻击的情况下，由入侵检测专家来判断是否存在一个成功的攻击。早期的入侵检测工具主要是通过人工从一大堆原始数据中提取数据，并对这些数据做智能的、合理的分析。现在的工具更加完善并且能够完成与其有相互关系的任务，不过从某一层面来说，入侵检测只是用工具来辅助，而不是由工具来直接操作，但仍然需要人来使数据相互关联。

现实中的很多攻击所利用的技术都是常用的方法集，它导致了相同的症状，但是几乎没有两个攻击是完全相同的。入侵检测工具检测这些常用的方法和特征，进而协助分析员进行入侵检测和分析。

在诸多入侵检测工具中，文件系统完整性检查工具主要监控计算机中的重要文件，当这些文件发生改变时，这些工具将生成报告。管理员就可以针对报告，检查值得怀疑的文件的非预期改变。例如，如果/etc/resolv.conf 或/etc/hosts 文件因为非可知原因发生了改变，那么通过文件系统完整性检测工具，管理员就能够知道该改变。也就是说这些工具用来检查文件的完整性。在检查过程中，比较一组文件或目录与以前在数据库中保存的信息是否一致，比较过程中发现的任何不一致（包括增加、删除和修改等）都被记录下来。当定期使用这些工具时，重要或被监控文件的任何改变可以被及时发现，因而可以立刻采用适当的控制措施来防止对系统进一步产生的危害。

目前最流行的两款文件系统的完整性检查工具是 aide 和 Tripwire，它们都在本书使用的系统中得到支持。用户可以从系统的软件源包安装它们，方法如下：

```
# yum install aide tripwire          # 红帽
# apt install aide tripwire          # Ubuntu
```

关于它们的具体用法这里不做进一步介绍。

10.3.2　事件报告制度

一个被捕获的异常访问企图、或者权限的提升，对一个系统或多个系统的机密、完整性或可访问性的入侵都可以称为一个事件。Linux 系统管理者应该把监控系统日志文件、系统完整性报告和系统记账报告当成日常工作。每个单位、每个部门或系统都应该有自己的事件报告制度，若发现问题或事件应该及时、慎重地上报。

1．为什么要报告事件

Linux 系统用户在使用系统的过程中要报告一个事件的原因有很多，下面列出了几个常见的原因。

（1）终止那些无休止的刺探行为。

（2）帮助保护其他站点。

（3）通知系统管理员。

（4）得到攻击的证实。

（5）让有关方面提高警惕和加强监控。

（6）安全制度的需要。

2．报告事件的类型

报告事件时，由用户决定报告哪些类型的事件。用户可以报告简单的端口扫描事件、拒绝服务攻击事件、访问秘密文件事件、非法占用系统资源事件等。

（1）企图获取登录系统账号的访问权。

常见的是刺探 TCP 端口 23 上的 telnet 服务、22 上的 ssh 服务，意图是执行命令并获得 shell 访问权。也可能是 21 号端口上的 ftp 服务，以获得系统的重要信息。

（2）企图重新配置系统。

如果要配置一个 Linux 操作系统，是需要有根用户权限的。一般情况下，攻击者无法获得根用户账号或密码，于是，入侵者就会想办法修改与用户或网络相关的数据，从而重新配置用户的系统，而达到某种目的。

（3）拒绝服务攻击。

所有拒绝服务的攻击都是恶意的，这些攻击一般以电子形式执行破坏、阻碍和盗窃服务等。

（4）企图访问秘密文件。

在 Linux 系统中有些文件是非公共文件，如用户账号密码文件、配置文件或用户私人文件等，攻击者非法访问这些秘密文件会对系统安全产生威胁。

（5）企图将文件存储在用户磁盘中。

这里指的是企图通过将不属于自己或不该自己掌握的数据占为己有，或用于其他目的。这种攻击是攻击者使用专门的盗窃软件，通过用户匿名 FTP 配置的漏洞将一些文件存储在用户磁盘中，也可能是偷偷地直接从本地系统复制。

3．向谁报告事件

当用户发现系统的潜在威胁时，或者已经发现系统受到攻击威胁时，应该把相关事件信

息报告给以下可能的部门：

（1）单位主管部门；

（2）发出进攻的站点的根用户或邮件管理员；

（3）网络管理员；

（4）ISP 运行中心；

（5）Linux 发行商。

4．事件报告应包含的信息

事件报告内容应该包含能够帮助事件响应或分析小组追查问题所需的足够信息，但是并不是说报告中包含的信息越多越好。用户还应该考虑所报告的信息是否含有隐私或是否会对自己的系统运行造成不便，例如，报告中可以包含用户的 E-mail 地址、对攻击的描述、对日志文件格式的描述、自己如何检测到该攻击和希望联系人做什么等信息，但不能包含与自己业务相关的客户及密码等信息。

10.4　防火墙

防火墙是一个或一组设备或软件，它在网络之间执行访问控制策略，是不同网络或网络安全域之间信息的唯一出入口。防火墙通过访问控制机制，确定哪些内部服务允许外部访问，以及允许哪些外部请求可以访问内部服务。防火墙虽有软件或硬件之分，但主要功能还是依据策略对外部请求进行过滤，成为公网与内网之间的保护屏障，防火墙会监控每一个数据包并判断是否有相应的匹配策略规则，直到满足其中一条策略规则为止，而防火墙规则策略可以是基于来源地址、请求动作或协议来定制的，最终仅让合法的用户请求流入内网中，其余的均被丢弃。

在红帽和 Ubuntu 系统中，均可以使用内核中内置的 iptables 作为防火墙数据包过滤机制，在红帽较新版系统中使用 firewalld 取代了 iptables 服务，在 Ubuntu 系统中既可以使用 firewalld，也可以使用 ufw 防火墙，但依然可以使用 iptables 命令来管理内核的 netfilter，例如，可使用 iptables-save 和 ip6tables-save 导出防火墙规则等。

10.4.1　iptables

iptables 是 Linux 的内核包过滤管理工具，它可以对系统内包过滤规则表格中的规则执行加入、插入或删除操作。iptables 工具作为软件包已经包含到 Fedora Core 的各版本中，并且作为系统的默认配置安装。

1．iptables 的链和策略

在 iptables 命令中设置数据过滤或处理数据包的策略称为规则，将多个规则合成一个链。数据包过滤防火墙中真正发挥过滤作用的是一系列过滤规则。作为内核包过滤管理工具，iptables 将对这些规则进行管理。iptables 常见的规则如表 10-1 所示。

表 10-1 iptables 常见规则

规　　则	描　　述
ACCEPT	允许通过
LOG	记录日志信息，然后传给下一条规则继续匹配
REJECT	拒绝通过，必要时会给出提示
DROP	直接丢弃，不给出任何回应
SNAT	伪装数据包的源地址，用于主机采用静态 IP 地址
DNAT	伪装数据包的目标地址

而规则链则依据处理数据包的位置不同而进行分类，如表 10-2 所示。

表 10-2 iptables 规则链

规　则　链	描　　述
PREROUTING	在进行路由选择前处理数据包
INPUT	处理入站的数据包
OUTPUT	处理出站的数据包
FORWARD	处理转发的数据包
POSTROUTING	在进行路由选择后处理数据包

iptables 中的规则表用于容纳规则链，规则表默认是允许状态，那么规则链就是设置被禁止的规则，而反之如果规则表是禁止状态，那么规则链就是设置被允许的规则。常见规则表如表 10-3 所示。

表 10-3 iptables 规则表

规　则　表	描　　述
raw	确定是否对该数据包进行状态跟踪
mangle	为数据包设置标记
nat	修改数据包中的源、目标 IP 地址或端口
filter	确定是否放行该数据包

2．iptables 命令及参数

iptables 的通用调用格式如下：

iptables [-t 规则表] 选项 [规则链] [条件] [-j 规则]

选项用来指定对链或规则的操作，包括插入、删除、清空规则等，常见选项如表 10-4 所示。

表 10-4 iptables 的常见选项

选　　项	描　　述
-L|--list [chain]	列出指定链上的所有规则。若未指定链，则列出所有链上的规则
-A|--append chain rules	在指定链规则末尾加入一条或多条规则

（续表）

选　　项	描　　述
-I|--insert chain [num] rules	在指定链加入一条或多条规则，若未指定编号 num 则默认头部
-R|--replace chain [num] rules	在指定的规则链中替换一条匹配的规则，若未指定 num 则默认 1
-D|--delete chain rules/num	在指定的规则链 chain 中删除一条或多条规则
-F|--flush [chain]	清空指定链中的所有规则
-N|--new-chain chain	新建一个用户自定义链
-E|--rename ochain nchain	重命名一个用户自定义链
-X|--delete [chain]	清除指定或所有的自定义链

3．Iptables 的地址转换

iptables 的地址转换功能主要是通过 NAT 实现的。

1）SNAT

SNAT 是源地址转换，应用于数据包即将被送出的前一刻，只有出站接口可以被指定，所以 SNAT 动作只在 POSTROUTING 规则链中使用。SNAT 的一般调用格式如下：

```
iptables -t nat -A POSTROUTING --out-interface 接口 … -j SNAT --to-source 地址[:端口]
```

其中，源地址可以映射到一个 IP 地址或一个可行的 IP 地址范围；源端口可以映射到路由器上的一个指定范围内的源端口。

2）DNAT

DNAT 是目的地址转换，应用于路由作出决定之前，所以 DNAT 动作可以在 OUTPUT 和 PREROUTING 规则链中使用。在 OUTPUT 规则链中，当指定出站接口时 DNAT 做其目标，调用语法如下：

```
iptables -t nat -A OUTPUT --out-interface 接口 … -j DNAT --to-destination 地址[:端口]
```

在 PREROUTING 规则链中，当指定入站接口时 DNAT 做其目标，调用语法如下：

```
iptables -t nat -A PREROUTING --in-interface 接口 … -j DNAT --to-destination 地址[:端口]
```

其中，目的地址可以映射到一个 IP 地址或一个可行的 IP 地址范围；目的端口可以映射到目的主机上的一个指定范围内的端口范围。

10.4.2　Firewalld

1．软件包安装与启动管理

红帽系列系统和 Ubuntu 均支持 Firewalld 防火墙。Firewalld 防火墙的软件包为 firewalld 及相关支持包，还有一个用于图形界面对 Firewalld 进行配置的工具包 firewall-config，可通过软件包管理工具安装它，安装命令如下：

```
#yum install firewalld firewall-config                  #红帽系统
#apt install firewalld firewall-config                  #Ubuntu
```

Firewalld 防火墙的服务为 firewalld.service，可使用 systemctl 管理它，方法如下：

```
#systemctl enable/disable/status firewalld.service          #启用/禁用/查询状态
#systemctl start/stop/restart/reload firewalld.service      #启动/关闭/重启/重载配置
```

2．相关概念

Firewalld 提供了动态管理的防火墙，支持网络/防火墙区域来定义网络连接或接口的信任级别。它支持 IPv4、IPv6 防火墙设置以及以太网桥接，并且拥有运行时配置和永久配置选项，还提供服务或应用的接口程序，可直接添加防火墙规则。

1）区域

防火墙的网络区域定义了网络连接的可信等级，数据包要进入内核必须要通过这些区域中的一个，而不同的区域里定义的规则不一样，即信任度不一样，过滤的强度也不一样。可以根据网卡所连接网络的安全性来判断网卡的流量到底使用哪个区域。一个网卡同时只能绑定到一个区域，可以把这些区域想象成进入火车站（地铁）的安检，不同的入口检测的严格程度不一样。

用户可根据系统使用的不同场景来使用不同的区域，Firewalld 预定义的区域及作用如表 10-5 所示，这些预定义的区域定义文件在/usr/lib/firewalld/zones 目录内。用户也可定义自己的区域，但要存放在/etc/firewalld/zones 目录内。

表 10-5　Firewalld 预定义的区域及作用

区　域	描　述
trusted	受信域。可接收所有的网络连接，允许所有的数据包
home	家庭域。拒绝外部连接，信任网内主机，只接收选定的外部连接
internal	内部域。等同于 home 区域
work	工作域。拒绝外部连接，信任网内主机，只接收选定的外部连接
public	公共域。不相信任何主机，允许选定连接
external	外部域。不相信任何主机，允许选定连接
dmz	非军事域。可受到公开访问且限制访问内部网络，只接收选定的传入连接
block	封锁域。拒绝任何外部连接，只有系统内启动的网络连接才是可能的
drop	丢弃域。丢弃所有外来包，只有向外部的连接是可能的

2）服务

对于服务大家比较熟悉，已经见于/etc/services，定义了服务名、使用端口号和协议等。这里的服务也是这个意思，被定义在/usr/lib/firewalld/services 中的服务有 100 多个。用户也定义自己的服务，但要存放在/etc/firewalld/services 目录内。

3．字符界面管理工具

Firewalld 提供图形和字符界面管理或配置工具。字符界面工具为 firewall-cmd 命令，通过该命令可以启用或关闭防火墙的相关特性，并可方便地添加、删除或修改相关防火墙规则。firewall-cmd 命令的常见子命令及意义如表 10-6 所示。

表 10-6　firewall-cmd 的常见子命令及意义

参　　数	作　　用
--get-default-zone	查询默认的区域名称
--set-default-zone=<zone>	设置默认的区域，永久生效
--get-zones	显示可用的区域
--get-services	显示预先定义的服务
--get-active-zones	显示当前正在使用的区域与网卡名称
--add-source=source	将来源于此 IP 或子网的流量导向指定的区域
--remove-source=source	不再将此 IP 或子网的流量导向某个指定区域
--add-interface=<interface>	将来源于该网卡的所有流量都导向某个指定区域
--change-interface=<interface>	将某个网卡与区域关联
--list-all	显示当前区域的网卡配置参数、资源、端口及服务等信息
--list-all-zones	显示所有区域的网卡配置参数、资源、端口及服务等信息
--list-service	显示允许的服务
--add-service=<service>	设置默认区域允许该服务的流量
--add-port=<port/protocol>	设置默认区域允许该端口的流量
--remove-service=<service>	设置默认区域不再允许该服务的流量
--remove-port=<port/protocol>	设置默认区域不再允许该端口的流量
--reload	让“永久生效”的配置规则立即生效，覆盖当前的规划
--state	查看防火墙状态

1）防火墙基本操作

```
##查看防火墙状态
#firewall-cmd --state
##显示所有预先定义的服务
#firewall-cmd --get-service
##显示当前域允许的服务
#firewall-cmd --list-service
##显示所有区域
#firewall-cmd --list-all-zones
##查看当前的区域
#firewall-cmd --get-default-zone
##查询 eth0 网卡的区域
#firewall-cmd --get-zone-of-interface=eth0
##在 public 中分别查询 ssh 服务是否被允许
#firewall-cmd --zone=public --query-service=ssh
```

2）配置实例

```
##设置默认规则为 dmz
#firewall-cmd --set-default-zone=dmz
##阻塞公共区域中的 echo-reply 报文
```

```
#firewall-cmd --permanent --zone=public --add-icmp-block=echo-reply
##启动应急状况模式，阻断所有网络连接
#firewall-cmd --panic-on
##允许 https 服务流量通过 public 区域，要求立即生效且永久有效
#firewall-cmd --permanent --zone=public --add-service=https; firewall-cmd --reload
##不再允许 http 服务流量通过 public 区域，要求立即生效且永久生效
#firewall-cmd --permanent --zone=public --remove-service=http; firewall-cmd -reload
##永久启用 home 区域中的 ipp-client 服务
#firewall-cmd --permanent --zone=home --add-service=ipp-client
##给 public 设置 source，来自 192.168.10.0/24 网段的都走 public 区
#firewall-cmd -zone=public -add-source=192.168.10.0/24
##允许 8080 与 8081 端口流量通过 public 区域，立即生效且永久生效
#firewall-cmd --permanent --zone=public --add-port=8080-8081/tcp; firewall-cmd --reload
##查看端口流量通过 public 区域中要求加入的端口操作是否成功
#firewall-cmd --permanent --zone=public --list-ports 8080-8081/tcp
##将 eth0 网卡的区域修改为 external，重启后生效
#firewall-cmd --permanent --zone=external --change-interface=eth0
#firewall-cmd --get-zone-of-interface=eth0
##将 home 区域的 ssh 服务转发到 127.0.0.2
#firewall-cmd --permanent --zone=home --add-forward-port=port=22:proto=tcp:toaddr=127.0.0.2
```

目前的 Ubuntu 和 Fedora 版本提供了一种不需要了解 iptables 语法，通过高级语言配置复杂 IPv4 和 IPv6 防火墙规则的机制。使用--add-rich-rule、--list-rich-rules、--remove-rich-rule 和 firewall-cmd 命令来管理它们。

```
##拒绝 192.168.10.0/24 网段的用户访问 ftp 服务
#firewall-cmd --permanent --zone=public --add-rich-rule="rule family="ipv4" source address="192.168.10.0/24" service name="ftp" reject"
```

4. 图形界面

Firewalld 图形界面配置工具是 firewall-config，启动方法是使用快捷键 Alt+F2，在对话框中输入 firewall-config 后回车，或直接从 CLI 界面执行命令：

```
#firewall-config
```

打开如图 10-1 所示的防火墙设置界面。

用户可以在图形界面选择区域，然后设置该区域的访问规则，很多原本复杂的长命令被用图形化按钮替代，设置规则也变得简单。

指定区域设置规则后，会在/etc/firewalld/zones 目录下生成“区域名.xml”文件，文件中既包含命令行配置的防火墙信息，也包括图形界面配置的防火墙信息。



图 10-1　Firewalld 配置界面

10.4.3　ufw

Ubuntu 系统既支持 firewalld 也支持 ufw 防火墙。ufw 防火墙软件包为 ufw，默认是安装的，但并不启用。若安装 ufw，可用以下命令：

```
#apt install ufw
```

用户可以通过命令 ufw help 得到简单的帮助，若使用命令 man ufw，则可得到稍详细的帮助。ufw 的常用方法如下：

```
ufw [options] enable|disable|reload                #启用|停用|重新加载
ufw [options] default allow|deny|reject [incoming|outgoing|routed]#允许|拒绝|拒绝并提示[进|出|路由]
ufw [options] logging on|off|LEVEL                 #打开|关闭|设置日志级别
ufw [options] reset                                #重置
ufw [options] status [verbose|numbered]            #显示状态，及规则
ufw [options] show REPORT                          #报告状态
ufw [options] [delete] [insert N] allow|deny|reject|limit [in|out] [log|log-all] [ PORT[/PROTOCOL] | APPNAME ] [comment COMMENT]   #[删除] [插入第 N 行] 允许|拒绝|拒绝并提示|限制 [入|出] [记录|全记录] [端口[/协议]]
ufw [options] delete N                             #删除第 N 行规则
ufw [options] app list|info|default|update         #应用列表|信息|默认策略|更新
```

选项有--dry-run：只显示应该的变化，但并不操作；-h/--help：显示帮助信息。

示例如下：

```
#ufw status                       #检查 ufw 状态
#ufw allow ftp                    #允许 ftp 服务
#ufw app list                     #查询应用列表
##允许来自 192.168.0.0-192.168.255.255 的数据通过 eth0 网卡进入主机
#ufw allow in on eth0 from 192.168.0.0/16
```

```
##允许指向 10.0.0.0-10.255.255.255 的数据通过 eth1 网卡从本机发出
#ufw allow out on eth1 to 10.0.0.0/8
##拒绝指向任意地址使用 80 端口 tcp 协议的数据进入本机
#ufw deny proto tcp to any port 80
##拒绝来自 10.0.0.0/8 域 tcp 协议指向 192.168.0.1 端口 25 的数据进入本机
#ufw deny proto tcp from 10.0.0.0/8 to 192.168.0.1 port 25
```

10.5　SELinux

SELinux（Security-Enhanced Linux）是美国国家安全局（NAS）对于强制访问控制的实现，在这种访问控制体系的限制下，进程只能访问那些在其任务中所需要的文件。大部分 Linux 系统，如 Fedora、RHEL、Centoe 和 Debian 等使用了 SELinux。本书使用的 Ubuntu 16.04 LTS，无论是桌面还是服务器版本，默认均未使用 SELinux，我们也遵守这个默认，因此所有与 SELinux 相关的内容均与 Ubuntu 无关。

SELinux 提供了一个可定制的安全策略，还提供很多用户层的库和工具，为系统的安全提供了保障。SELinux 的应用特点包括以下几方面。

1．MAC（Mandatory Access Control）

MAC 为强制性访问控制，做到对访问的控制彻底化。对所有的文件、目录、设备、端口等的访问都是基于策略设定的，这些策略由管理员设定，一般用户无权更改。

2．TE（Type Enforcement）

TE 为类型强制，对于进程只赋予最小的权限。对所有的文件、目录、设备等对象都赋予一个被称作 type 的类型标签，对于每个进程也都赋予一个叫域（domain）的标签或属性。domain 标签能够执行的操作也是在策略里定义好的。每个用户都基于他的操作域及访问控制策略来访问系统中的客体对象。

3．RBAC（Role-Base Access Control）

RBAC 为基于角色的访问控制，对用户只赋予最小访问权限。用户被划分成了一些角色（role），角色按层次分类，权限按角色分配，用户按照自己的角色和权限分配策略行使权力。即使是 root 用户，如果不具有 sysadm_r 角色的话，也不能执行相关的管理工作。但是，哪些 role 可以执行哪些 domain，也是可以修改的。

4．安全上下文

当启动 SELinux 的时候，所有文件等对象都有安全上下文，进程的安全上下文是域，安全上下文由“用户：角色：类型”表示。

10.5.1　SELinux 中的安全类型和角色

所有操作系统访问控制都是以关联的客体和主体的某种类型的访问控制属性为基础的。在 SELinux 中，访问控制属性称为安全上下文。所有客体（文件、进程间通信通道、套接字、

网络主机等）和主体（进程）都有与其关联的安全上下文，一个安全上下文由三部分组成：用户、角色和类型标识符。

1．类型

在 SELinux 中，类型标识符是安全上下文的主要组成部分，当一个类型与执行的进程关联时，一个进程的类型通常被称为一个“域”或“域类型”。从本质上讲，类型（组）主要是用来将主体与客体划分为不同的组，而组的每个主体和系统中的客体定义了一个类型；类型为进程运行提供最低的权限环境。域或安全上下文是一个进程允许操作的列表，决定一个进程可以对哪种类型进行操作。常见的类型有 sysadm_t（具有特权的管理角色和用户域类型）、user_t（普通用户类型）、staff_t（特权用户的非信任域类型）和 unconfined_t（非制定类型）。对于网络服务等，比如 httpd 和 squid 等，还有 httpd_t 和 squid_t 等类型。

2．角色

SELinux 提供了一种依赖于类型强制基于角色的访问控制，角色用于组域类型和限制域类型与用户之间的关系。SELinux 中的每个用户关联一个或多个角色，使用角色和用户，RBAC 特性允许有效地定义和管理最终授予 Linux 用户的特权。

SELinux 中的角色允许限制用户的访问，对于任何进程，同一时间只有一个角色（即进程安全上下文中的角色）是活动的。简单地说，可以为角色指定不同的权限，不同的用户有不同的角色，只有切换到特定的角色才有特定的权限。

SELinux 中的常见角色如下：

（1）文件与目录的角色，通常是 object_r；

（2）程序（进程）的角色，通常是 system_r；

（3）用户角色，在 targeted 策略环境中用户的角色一般为 system_r。在 strict 策略中用户的角色被细分为 sysadm_r，system_r，user_r，staff_r。用户的角色类似组的概念，不同的角色具有不同的身份权限。一个用户可以具备多个角色，但是同一时间只能使用一个角色。在 targeted 策略环境下角色没有实质作用，所有进程文件的角色都是 system_r。在 strict 策略环境中，一般用户登录时角色是 user_r，root 登录时角色是 staff_r，但没有特权，切换到 sysadm_r 角色才具备超级用户的权限，同时只有 staff_r 角色可以切换成 sysadm_r。

3．SELinux 中的用户

SELinux 中的用户（User Identity）类似 Linux 系统中的 UID，主要提供身份识别，是安全上下文中的一部分。在启用 SELinux 的系统上，SELinux 用于定义用户权力的模型和已有的 UNIX 用户模型是共存的。一个在 SELinux 中还分配了角色的用户称为定义用户，这种用户有三种：

（1）root：用来与传统 UNIX 的 root 用户标识的上下文兼容。

（2）system_u：开机过程中系统进程的预设，或者说是分配给系统进程的用户标识。

（3）user_u：普通用户标识。

所有 SELinux 的预设用户均以“_u”结尾，但 root 用户除外。通过用户可以确定身份类型，但在 SELinux 中用户一般搭配角色使用，身份和不同的角色搭配时形成不同权限。

10.5.2　SELinux 中的策略

策略就是可以设置的规则，决定一个角色的用户可以访问什么，哪个角色可以进入哪个域，哪个域可以访问哪个类型等问题。用户可以根据想要建立的系统特点来决定设置什么样的策略。

SELinux 目前有三个安全策略：targeted，strict 和 mls。

（1）targeted：用来只保护选定的网络服务进程，此为 SELinux 的默认值。

（2）strict：用来保护整个系统。

（3）mls：用来提供符合 MLS（Multi-Level Security）机制的安全性。

10.5.3　SELinux 的 Targeted 策略

在 Fedora 中 Targeted 是默认策略，只用来保护网络服务。当使用 Targeted 策略时，有 Targeted 标识的进程工作在 confined 域，其他进程工作在非限制域（Unconfined Domain）。比如，登录的用户或进程工作在 unconfined_t 域，由系统初始化进程 init 启动的进程工作在 initrc_t 域，且这两个域均是 unconfined 的。

SELinux 仅为网络服务提供最低需要的访问权限，但是，网络服务可以以多种方式运行，管理员必须告诉 SELinux 如何运行指定的网络服务，这些可以通过一些相关的布尔变量来控制（见后述）。

1．TE

TE 是 SELinux Targeted 策略下的主要存取允许控制策略。在此方式下，每个文件和进程都指定一个类型标识，该类型标识为进程定义了一个域，为文件定义了一个类型，策略规则定义了类型间如何相互访问，访问可以是域访问类型，或域访问域。这种访问只在两者之间有允可规则存在时才被允许。

2．限制进程（Confined Processes）

在几乎所有支持 SELinux Targeted 策略的系统中，每个在网络上进行监听的服务进程都是限制性的，同样所有以 root 用户身份工作的进程和为用户执行任务的进程也是限制性的。当一个进程是限制性的时，它工作在自己的域中，比如，httpd 工作在 httpd_t 域。如果一个限制性进程被攻击或被攻破，则按照 SELinux 的规则设置，攻击所能造成的破坏或可以访问资源的范围也是会被限制的。

3．非限制进程（Unconfined Processes）

非限制进程工作在非限制域。例如 init 进程工作在非限制域 initrc_t，系统内核进程工作在非限制域 kernel_t，非限制性用户工作在非限制域 unconfined_t。对于非限制性进程，SELinux 策略也是被应用的，但对于在非限制域上存在允许策略者是不被禁止的。进程工作在非限制域时，访问控制回退到 DAC 规则。假如，此时非限制进程被攻破，SELinux 并不阻止攻击者获得系统资源、数据和访问权。

10.5.4　SELinux 的策略及改变

1．SELinux 的配置文件

SELinux 的配置文件为/etc/selinux/config，其中定义了系统启动时为 Fedora 所使用的策略。配置文件的有效内容为：

```
#This file controls the state of SELinux on the system.
#SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
#SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

其中

```
SELINUXTYPE=targeted
```

定义了系统启动时所使用的默认策略为 targeted。在 Fedora 中 SELINUXTYPE 只能取 targeted 或 mls。若要使用 mls 还需要扩充安装 selinux-policy-mls 及相关软件包。

配置文件中还有一个关键变量 SELINUX（SELinux 的状态），可以取三个值：

enforcing：打开当前策略，并且实施这个策略。

permissive：打开当前策略，但并不真正实施，只是显示警告信息。

disabled：关闭 SELinux。

2．SELinux 的策略修改

有多种办法可以修改默认策略，比如可以通过修改 SELINUXTYPE 的值来改变默认策略，也可以通过命令 chcon 来修改相关的具体策略值（见后述）。通过命令：

```
#setenforce 1/enforcing          #切换为 enforcing 方式，或
#setenforce 0/permissive         #切换为 permissive 方式
```

可以临时在 enforcing 和 permissive 方式之间进行切换。如果想关闭 SELinux，则只能通过修改/etc/selinux/config 配置文件来实现。

若在当前配置（SELINUX=enforcing）下想使用 permissive 策略，可在启动时修改启动参数，在 GRUB 界面的行末添加以下内容：

```
setenforce 0
```

3．查询 SELinux 的当前状态

命令 setenforce 用于设置 SELinux 的工作状态，而 getenforce 可用于查询 SELinux 的当前

状态，在终端屏幕上显示当前状态描述串“enforcing”或“permissive”。

若要查询 SELinux 是否开启，则需要使用命令 selinuxenabled，该命令不显示任何输出，只给出一个返回码，可用命令“echo $?”观查其结果，为 0 时表示开启，否则表示未开启。

10.5.5　SELinux 中的布尔值及改变

SELinux 中有很多布尔型变量，它们的值可以决定是否打开或关闭对某个目标的保护，当一个布尔变量的值为真（true 或 1）时，表示保护，为假（false 或 0）时表示不保护。SELinux 的保护策略为每个布尔变量设定了一个默认值，这个值通常为 false。

1．查看布尔变量及其值

可以通过命令 getsebool 得到所有或指定布尔变量名和值。getsebool 的用法为：

```
getsebool [-a] [bool_var … ]
```

bool_var 为某个布尔变量名，用于指定布尔变量；-a 用于显示所有布尔变量及其值，比如显示所有布尔变量及其值可以使用命令：

```
getsebool -a
```

用于显示所有与 ftp 相关的布尔变量及其值可以使用命令：

```
getsebool -a | grep ftp
```

2．改变布尔变量的值

还可以通过 setsebool 来设置或改变布尔变量的值，从而改变 SELinux 对某个目标的保护状态。setsebool 的用法如下：

```
setsebool [ -P ] boolean value
setsebool [ -P ] bool1=value bool2=value ...
```

value 的值可以是 1、true 或 on 用于表示打开或启用，也可以是 0、off 或 false，表示关闭或不启用。参数-P 用于决定是临时或永久改变某个布尔变量的值，若不使用-P，则改变只在当前有效，在系统重启后恢复为原来的值；若使用-P 参数，则修改值将存到磁盘，在下次系统重启后仍然有效。

命令 togglesebool 可以用来使布尔变量的值在真假之间转换，其用法为：

```
togglesebool bool_var1 …
```

10.5.6　检查 SELinux 的状态

可以通过命令 sestatus 检查 SELinux 的状态，检查内容包括 SELinux 是否开启，默认策略，以及采用 enforcing 或 permissive 方式，还可以报告/etc/sestatus.conf 内文件和进程的安全上下文。sestatus 的用法为：

sestatus [-b] [-v]

参数-v 用于追加报告/etc/sestatus.conf 中文件和进程的安全上下文，如果是符号链接，也将报告所链接目标的情况。-b 用于追加报告所有布尔变量的当前值，等价于 getsebool -a。

```
#sestatus
    SELinux status:              enabled          #当前状态
    SELinuxfs mount:             /selinux         #SELinux 文件系统安装目录
    Current mode:                enforcing        #当前方式
    Mode from config file:       enforcing        #配置文件默认方式
    Policy version:              22               #当前版本
    Policy from config file:     targeted         #当前策略
```

执行命令：

```
#setenforce 0
```

之后，再执行 sestatus 可以到输出中的 Current mode 内容应为：

```
Current mode:          permissive
```

而其他内容保持不变，说明当前模式变为了 permissive。

10.5.7　SELinux 安全上下文管理

1．查看安全上下文命令

1）检查文件与目录的安全上下文

```
#ls -dZ /var/ftp /var/www                  #查看/var/ftp 和/var/www 的安全上下文
drwxr-xr-x … system_u:object_r:public_content_t:s0 /var/ftp
drwxr-xr-x … system_u:object_r:httpd_sys_content_t:s0 /var/www
```

2）检查进程的安全上下文

```
#ps -eZ|grep -E '（ftp）|（name）d'         #查看 named 和 vsftpd 的安全上下文
system_u:system_r:named_t:s0     2034 ?        00:00:00 named
system_u:system_r:ftpd_t:s0      2205 ?        00:00:00 vsftpd
```

3）检查账号的安全上下文

```
#id -Z                                     #查看当前用户的安全上下文
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

2．安全上下文管理

1）修改文件对象的安全上下文
在 SELinux 中 chcon 用于修改文件对象安全上下文，其用法为：

```
chcon [OPTION] … CONTEXT FILE...
chcon [OPTION] … [-u USER] [-r ROLE] [-l RANGE] [-t TYPE] FILE...
chcon [OPTION] … --reference=RFILE FILE...
```

常用参数或选项如表 10-7 所示。

表 10-7　chcon 的常用参数

参　数	意　义	参　数	意　义
-v	显示冗长的信息	-c	像-v，但只在修改时报告
-R	递归地修改对象的安全上下文	-h	不跟踪符号链接
--reference=FILE	通过 FILE 指定一批文件	-t TYPE	修改安全上下文类型
-u USER	修改安全上下文用户	-r ROLE	修改安全上下文角色

2）修改文件对象默认安全上下文

在 SELinux 中 restorecon 用于恢复文件对象的安全上下文，其用法为：

```
restorecon [-o outfile] [-R] [-n] [-v] [-e dir] pathname ...
restorecon -f infile [-o outfile] [-e dir] [-R] [-n] [-v] [-F]
```

常用参数或选项如表 10-8 所示。

表 10-8　restorecon 的常用参数

参　数	意　义	参　数	意　义
-v	显示改变信息	-F	对用户自定义文件强制恢复
-R	递归处理目录	-n	不做恢复
-f FILE	通过 FILE 指定一批文件	-o FILE	将安全上下文不正确的都保存到 FILE
-e dir	不包含目录 dir	pathname	指定目标

3）示例

以下为文件安全上下文的修改与恢复过程：

```
#ls -d -Z /var/ftp/pub
    drwxr-xr-x   root root system_u:object_r:user_home_t:s0 /var/ftp/pub
#chcon -t var_t /var/ftp/pub; ls -Z /var/ftp/pub
    drwxr-xr-x   root root system_u:object_r:var_t:s0        /var/ftp/pub
#restorecon   -v /var/ftp/pub
    restorecon   reset   /var/ftp/pub   context   system_u:object_r:var_t:s0->
system_u:object_r:user_home_t:s0
#ls -d -Z /var/ftp/pub
    drwxr-xr-x   root root system_u:object_r:user_home_t:s0 /var/ftp/pub
```

10.5.8　SELinux 的应用

1．网络服务所使用的 SELinux 文件类型

在 Linux 中，SELinux 使用的是 Targeted 策略，主要用于保护 Linux 系统的网络服务。如果欲使自己的 Linux 系统在网络上安全运行，就必须开启 SELinux。可以通过命令：

```
#getenforce
```

确认自己的系统是否工作在 SELinux 的保护之下，若不是这样，先开启 SELinux。

可以通过命令：

```
#semanage port -l                    #显示所有受保护的服务和端口号
```

查看所有受保护的服务和端口号；也可以通过命令：

```
#semanage port -l | grep SERVER      #SERVER 为服务名或端口号，比如 ftp，telnet 等
```

查看受 SELinux 保护的指定服务端口。

SELinux 对网络服务的保护除了依靠布尔变量的值外，还与被访问文件的安全上下文有关。SELinux 默认是对网络服务不开放的，如果需要开放就要使用 setsebool 命令将对应的布尔变量的值置为 on。

对于每种服务中所需要设置的文件的安全上下文类型则各不相同，需要针对不同服务单独设置，可参考其版本，比如 Fedora 24 的《SELinux User's and Administrator's Guide》。文件 /etc/selinux/targeted/contexts/files/file_contexts 内包含了系统内所有目录和文件的 SELinux 文件类型，本书所涉及服务的目录或文件的类型标签及意义如表 10-9 所示。

表 10-9　SELinux 的目录或文件的类型标签及意义

类　型	意　义	服　务
public_content_t	vsftpd 只读共享的文件类型	vsftpd
public_content_rw_t	vsftpd 读写共享文件类型	vsftpd
var_lib_nfs_t	/var/lib/nfs/及其内文件类型	nfs
nfsd_exec_t	与 nfs 相关的可执行文件类型	nfs
named_zone_t	DNS 主区文件类型，其他服务不能修改	named
named_cache_t	DNS 缓存文件类型，named 可写	named
named_var_run_t	/var/run/下的 bind、named 和 unbound 目录及其下文件	named
named_conf_t	named 相关的/etc/下的目录或文件	named
named_exec_t	/usr/sbin/下与 BIND 相关的可执行文件	named
named_log_t	/var/log/下与 BIND 相关的 log 文件	named
named_unit_file_t	/usr/lib/systemd/system/下与 BIND 相关的文件	named
httpd_sys_content_t	静态页面文件类型，只读访问	httpd
httpd_sys_script_exec_t	httpd 可执行脚本文件类型	httpd
httpd_sys_rw_content_t	可被 httpd 可执行脚本读和写的文件类型	httpd

（续表）

类　　型	意　　义	服　务
httpd_sys_ra_content_t	可被 httpd 可执行脚本读和追加的文件类型	httpd
httpd_unconfined_script_exec_t	不受 SELinux 保护的 httpd 脚本类型	httpd
samba_share_t	samba 共享文件类型	samba
samba_etc_t	/etc/samba/目录及其内文件类型	samba
dhcp_etc_t	/etc/dhcp/目录及其内文件类型	dhcp
dhcpd_var_run_t	/var/run/下的 dhcpd PID 文件	dhcp
dhcpd_exec_t	向 dhcpd_t 域传送 DHCP 可执行文件	dhcp
dhcpd_initrc_exec_t	向 dhcpd_initrc_t 域传送 DHCP 可执行文件	dhcp

2．SELinux 与网络服务关系示例（ftp）

我们以 SELinux 与 vsftpd 的关系为例，说明如何处理 SELinux 与网络服务的关系。

1）SELinux 布尔变量处理

在 SELinux 开启的情况下，当一个本地用户成功登录 vsftp 后，默认不能访问自己的家目录，vsftpd 不能访问 NFS 和 CIFS 文件系统，匿名用户不具有写权限，即便是配置文件 vsftpd.conf 允许也不行。但是通过将一些与 vsftpd 相关的布尔值置 1（true），可以解决以上问题，方法如下所述。

首先执行命令：

```
#getsebool -a | grep ftpd
```

得到与 ftpd 相关的布尔变量及其状态。

```
ftpd_anon_write --> off                    #匿名用户上传功能关闭
ftpd_connect_all_unreserved --> off        #其他未预定域连接关闭
ftpd_connect_db --> off                    #连接数据库关闭
ftpd_full_access --> off                   #ftpd 全部访问控制关闭
ftpd_use_cifs --> off                      #访问 cifs 文件系统关闭
ftpd_use_fusefs --> off                    #访问 fusefs 文件系统关闭
ftpd_use_nfs --> off                       #访问 nfs 文件系统关闭
ftpd_use_passive_mode --> off              #ftpd passive_mode 关闭
```

其次，使用命令 setsebool 开放或永久开放所需要的布尔变量，比如只开放 ftpd 服务，则需要执行命令：

```
#setsebool allow_ftpd_full_access 1        #允许 ftp 服务器访问系统的任何地方
#setsebool -P allow_ftpd_full_access 1     #永久允许 ftp 服务器访问系统的任何地方
```

若需要开放匿名用户上传，则需要执行命令：

```
#setsebool allow_ftpd_anon_write 1         #开启匿名用户上传功能
#setsebool -p allow_ftpd_anon_write 1      #永久开启匿名用户上传功能
```

2）SELinux 文件类型处理

应该注意，在 SELinux 的管理下，有时候只有 Linux 系统的权限和 vsftd 配置文件赋予的权限还可能是不够的，必要时还需要对特定文件或目录赋予 SELinux 的类型权限。可以使用

命令 chcon 或 semanage 来进行。

现在我们看一看目录/var/ftp 和/var/ftp/incomg 的 SELinux 权限：

```
#ls -Z -d /var/ftp /var/ftp/incoming
    drwxr-xr-x    root root system_u:object_r:public_content_t:s0 /var/ftp
    drwxrwxr-x    ftp ftp unconfined_u:object_r:public_content_t:s0 /var/ftp/incoming
```

由此可见，incoming 目录的系统权限是 drwxrwxr-x，且主、组均为 ftp，如果不考虑 SELinux，文件上传是没有问题的。但是，它们的 SELinux 类型都是 public_content_t，只能以只读方式使用，不能写。如果要让匿名用户向/var/ftp/incoming 内上传文件，还必须修改它的 SELinux 文件类型，从原来的 public_content_t 修改为 public_content_rw_t，方法是：

```
#chcon -R -t public_content_rw_t /var/ftp/incoming
```

此时，其 SELinux 文件类型为：

```
unconfined_u:object_r:public_content_rw_t:s0 /var/ftp/incoming
```

之后，匿名用户就可以向/var/ftp/incoming 目录上传文件了。对于其他目录，若需要，可以仿此处理。

习题 10

1．思考题

（1）试述 Linux 系统的安全机制及安全防范策略。
（2）试述 firewalld 的区域。
（3）试述 SELinux 的常用命令及作用。
（4）试述事件报告制度。

2．单项选择题

（1）下面哪一个不是 SELinux 上下文的构成（　　）？
A．用户　　B．角色　　C．防火墙　　D．类型
（2）下面哪一个不是 iptables 的规则表（　　）？
A．nat 表　　B．filter 表　　C．mangle 表　　D．INPUT 表
（3）SELinux 中用于修改上下文的命令是（　　）。
A．chcon　　B．setsebool　　C．setenforce　　D．umount
（4）在 Linux 下使用哪个命令可以显示系统网卡的 IP 地址及其他网络设置（　　）？
A．ifconfig　　B．ps　　C．netstat　　D．route

实验 10

1．针对所用系统，使用 firewalld 或 ufw 的配置工具配置防火墙。
2．若为红帽系统，使用相关命令检查 SELinux 的状态、布尔值和安全上下文。

编程与开发篇

- bshell 编程
- C/C++及其他编程环境

第 11 章　bshell 编程

11.1　正则表达式

正则表达式是由一个或多个字符或元字符组成的串。普通字符包括大、小写字母和数字，而元字符则是具有特殊意义的字符。支持模式搜索的命令一般都支持正则表达式。正则表达式在不同的系统中的实现可能是不同的。

一个正则表达式包含以下内容。

（1）字符集：在指定位置上匹配的一个或多个字符；

（2）计数：指定其前面的字符重复的次数，比如“*”、“?”、“+”；

（3）位置字符：特殊字符集，用于标明位置，比如行首、行末等；

（4）具有特殊意义的字符。

有三种形式的正则表达式：

（1）SRE（simple regular expression）：传统 UNIX 支持这种正则表达式；

（2）BRE（basic regular expression）：ISO POSIX-2 支持这种正则表达式；

（3）ERE（extended regular expression）：ISO POSIX-2 也支持这种正则表达式。

ERE 与 BRE 在用法上不尽相同，且 ERE 比 BRE 有更丰富的语法，但是 ERE 不是 BRE 的超集，一些 BRE 的表达式若不加修改，则在 ERE 中不能正常使用。在 Linux 系统中 ARE（Advanced Regular Expression）是对 ERE 的扩展，ERE 的表达式可直接在 ARE 上使用。

在有些 UNIX 系统中 awk，egrep 使用 ERE，而其他程序（比如，ed，ex，expr，grep，sed 和 vi）使用 BRE。

11.1.1　字符集

1．元字符

元字符是指具有特殊功能的字符，它在正则表达式中不代表自己，而具有特殊的意义。常用的元字符及意义如表 11-1 所示。

表 11-1　元字符及其意义

元　字　符	意　　义	示　　例
.	匹配除换行后以外的其他任意字符	a.c 匹配 abc,adc 等
*	匹配 0 或多个前面刚出现的字符或正则表达式	aa*匹配 a,aa,aaa,…
?	匹配其前面正则表达式的 0 次或 1 次出现（ERE）	bb?匹配 b 或 bb
+	匹配其前面正则表达式的 1 次或多次出现（ERE）	(ab)+匹配 ab,abab,…
^	位置字符表示行首	^abc 匹配行首为 abc 的行
$	位置字符表示行末	ter$匹配行末为 ter 的行

（续表）

元 字 符	意 义	示 例
[ ],[^]	匹配或不匹配其中的一个字符。如果在选择集中包含[]，则它必须是第一个。^为[ ]内首字符，表示相反的操作	[Bb]y 匹配 By 或 by。a[^Bb]c 不匹配 abc 或 aBc
|	从多个选项选择（ERE）	((auto)|(dog)matic)匹配 automatic 或 dogmatic
\	转义字符；将其后面的一个字符变为普通字符	\”匹配字符”
( )	BRE 组或子表达式。定义在圆括号的字符串以备后用。以后 1～9 表示前面保存的正则表达式	((auto)|(dog)matic)匹配 automatic 或 dogmatic
\(\)	ERE 组或子表达式。保存在圆括号的字符串以备后用。以后\1 到\9 表示前面保存的正则表达式	\(more\) and \1 匹配 more and more
{l}，{l,},{l,u}	匹配指定数量的重复，l 为下界，u 为上界（BRE）	(is{2}){2}:ississ
\{l\},\{l,},\{l,u\}	匹配指定数量的重复，l 为下界，u 为上界（ERE）	(ab)\{2\}匹配 abab
\<	在一个字的开始匹配其后正则表达式	\<exp 匹配以 exp 开头的词
\>	在一个字的结尾匹配其前的正则表达式	ble\>匹配以 ble 结尾的词
[. .]	组合类。匹配多个字符组合；[[.ijx.]]不同于[ijx]	[[.ijx.]]匹配 ijx
[= =]	相等类。匹配相同字符	[d[=e=]f]：[deeeef]
[: :]	类字符（参见表 11-2）	[:digit:]定义数字类

2．类字符

类字符用于表示某一种类型的字符（集），可以用来简化正则表达式的长度，类字符如表 11-2 所示。

表 11-2　类字符

类 字 符	意 义	示 例
[:alnum:]	字符和数字	0～9,a～z,A～Z
[:alpha:]	字符	A～Z,a～z
[:blank:]	白空格。POSIX 只包括空格和 tab 键	空格键、tab 键等
[:cntrl:]	控制字符	
[:digit:]	数字	0～9
[:xdigit:]	十六进制字符	0～9,a～f
[:graph:]	制表符	
[:lower:]	小写字符	a～z
[:upper:]	大写字符	A～Z
[:print:]	可打印字符	
[:punct:]	标点符号	.,?,!等

3．BRE 使用的字符集

BRE 字符集是指用于 BRE 正则表达式的特殊字符集，或在 BRE 正则表达式中具有特殊意义的字符集。BRE 使用的特殊字符集如表 11-3 所示。

4．ERE 匹配字符集

ERE 字符集是指用于 ERE 正则表达式的特殊字符集，或在 ERE 正则表达式中具有特殊

意义的字符集。ERE 使用的特殊字符集如表 11-4 所示。

表 11-3　BRE 使用的字符集

元　字　符	意　　义	元　字　符	意　　义
[=,=]; [:,:]; [., .]	等价类、字符类和联合类	\(,\),\#	子表达式，后向引用
\character	（通过\实现）特殊字符	*,\{l\},\{l,\},\{l,u\}	匹配 0 个或多个
[ ]	括号表达式	^, $	位置字符

表 11-4　ERE 使用的字符集

元　字　符	意　　义	元　字　符	意　　义
[=,=]; [:,:]; [., .]	等价类、字符类和联合类	*,+,?,{l},{l,}{l,u}	匹配 0 个或多个
\character	（通过\实现）特殊字符	^,$	位置字符
[ ]	方括号表达式	|	多种选择
( )	组表达式		

11.1.2　shell 正则表达式

shell 正则表达式和正则表达式在某些方面存在不同。例如 case 模式匹配的是 shell 变量，文件内容模式匹配与搜索时，应用程序 ex、vi、view、grep、sed、awk 等有的用的是 BRE，有的用的是 ERE，要视具体情况而定。用于 shell 的正规表达式元字符如表 11-5 所示。

表 11-5　用于 shell 的正规表达式元字符

元　字　符	意　　义
?	匹配任意字符
*	匹配 0 个或多个字符
\character	通过转义字符实现对特殊字符的匹配
[ ], [!...]	与正则表达式相同
{expr1,expr2, …}	匹配{ }中的任意字符串

说明：这里主要指的是通配符或文件名扩展或叫路径扩展。比如，“a {user,group,other} is”分别匹配“a user is”、“a group is”和“a other is”，其他的示例参见 3.1.2 节通配符部分。

11.2　流编辑（sed）

11.2.1　功能及用法

sed 是个流编辑器，其功能是对它的输入流进行按指定格式的转换。sed 使用脚本语言对输入流进行加工，脚本可以在命令行或文件中输入。在工作时它只对输入流处理一遍，但却效率非常高。为了能充分利用 sed 的功能和效率，可以进行脚本编程，其用法为：

```
sed [options] { scripts } [input-file] …
```

11.2.2　参数与说明

sed 的部分参数如表 11-6 所示。

表 11-6　sed 的部分命令行参数

参　　数	描　　述
-n,--quiet,--silent	抑制不必要的输出，若不使用-n，则输出操作过程的每一行
-e script,--expression= script	在命令行加入脚本
-f script-file	从指定文件中读入脚本
-r,--regexp-extended	使用扩展的正则表达式

在 sed 的命令行中，若没有使用-e 或-f 指定脚本，则它的第一个非选项变量将作为脚本，而其他的变量将作为输入文件，若无输入文件，则使用标准输入。sed 的输出对象为标准输出，若输出到文件可使用输出重定向。

11.2.3　脚本命令

1．sed 的地址形式

在编写脚本程序时，要用到地址范围的概念。sed 的地址范围可以表示为如表 11-7 所示形式。

表 11-7　sed 的地址及地址范围形式

地 址 范 围	意　　义	地 址 范 围	意　　义
begin_num～step	从 begin_num 开始每 step 行执行一次	$	最后一行
L1,L2	从 L1 到 L2 行	L1,+N	从 L1 到 L1+ N 行
L1,～N	从 L1 到行号为 N 的一个倍数的行	/regexp/	与模式 regexp 匹配的行

2．sed 脚本中的部分命令

sed 之所以能够处理脚本，是因为它有自己的内部命令。sed 的部分内部命令如表 11-8 所示。

表 11-8　sed 的部分内部命令

命　　令	功　　能	命　　令	功　　能
;	脚本命令分隔符	#	注释
{ }	命令组	:lbl	标号，设置跳转目标
=	显示当前行行号	d	删除指定行（不影响原输入文件的内容）
l	显示当前行内容	q [code]	立即退出 sed（返回码为 code）
p	显示模式空间内容（一般与-n 配合使用）	!func	不执行 fun（func 的值有:=,l,d 和 p 等）
R/r file	从文件中追加文本/行	wfile	将当前行内容写入文件 file

（续表）

命　　令	功　　能	命　　令	功　　能
T/t/b [lbl]	跳转到标号 lbl，若不指定则跳转到脚本末	i\text	在指定行前插入 text 文本
a\text	在指定行后追加 text 文本	c\text	以字符串 text 取代当前行
s/rexp1/rexp2/flag	查找 rexp1 并用 rexp2 替换。flag 为：*n*-整数，只替换第 *n* 次出现；g-全部；p-如果替换，则显示；wfile-如果替换，则写入文件 file	y/src/dest/	在指定范围内用 dst 内的字符集对应替代 src 的对应位置的字符。dst 与 src 的字符个数必须相等

11.2.4　sed 示例

（1）将 ifile 文件中的所有 sysman 替换成 System Manager 并输出到 ofile。

```
#sed -e 's/sysman/System Manager/g' <ifile >ofile
```

（2）计算件文件 ifile 的行数。

```
#sed -n '$=' <ifile
```

（3）删除文件 ifile 中的空行，并将其他的内容输出到 ofile。

```
#sed '
   /^$/d
   /^[:space:]*$/d
   ' <ifile >ofile
##也可写为：
#sed '/^$/d; /^[:space:]*$/d' <ifile >ofile          #或
#sed '/^$/d; /^[:blank:]*$/d' <ifile >ofile
```

（4）显示 ifile 文件中的所有注释行/非注释行。

```
#sed -e '/^#/!d' < ifile                        #显示注释行。参见不执行某命令!func
#sed -e '/^#/d' < ifile                         #显示非注释行
```

（5）报告系统中所有（非）root 用户的活动。

```
#ps -ef | sed -e '/^[:space:]*root/!d'          #显示系统中 root 用户进程
#ps -ef | sed -e '/^[:blank:]*root/d'           #显示系统中的非 root 用户进程
```

（6）去除一个文本文件的所有行首和行末的白空格。

完成此任务可以使用以下脚本（_为空格，为可视而设）：

```
y/\t/_/
s/^_*//
s/_*$//
```

如果认为内容较多，可以将其存入脚本文件，如 efile.sed 中，此时命令的执行方法为：

```
#sed -f efile.sed < ifile > ofile
```

如果再在 efile 文件的首处为其增加一“标识”行：

```
#!/bin/sed -f
```

则可以在为其添加执行权后直接执行之。操作过程如下：

```
#chmod +x efile.sed
#./efile.sed < ifile > ofile
```

11.3 模式搜索与处理（awk）

11.3.1 功能及用法

awk 是一个用于模式搜索和处理的编程语言。它最早的设计目的是针对报表生成的一种小巧且具表达力的语言，因此 awk 对于格式化结构的文件特别有效，其用法为：

```
awk [options] -f progfile [--] file …
awk [options] [--]'program' file …
```

11.3.2 参数说明

awk 有很多参数用于工作过程的控制，其部分参数如表 11-9 所示。

表 11-9 awk 的部分命令行参数

参　　数	描　　述
-f progfile	指定 awk 程序文件
-F fs	设置格式化文件的记录中域间分隔符
-v var=value	定义并指定变量的值

11.3.3 记录和域

awk 记录和域的概念与数据库里记录和域的概念类似。在一个格式记录的文件里，它的每一行称为一个记录，而每一个记录又由域分隔符分隔的不同字段组成，这种字段就是记录中的域。在 awk 中整个记录用$0 表示，而组成记录的每个域依次表示为$1，$2……

11.3.4 变量

awk 是可以编程的程序或语言，自然离不开变量。用户在编程时可以像其他语言一样定义用户变量。在 awk 中变量是动态定义，只有当它第一次被使用时变量才存在。由于是解释语言，变量可以是任何类型的，这取决于用户如何使用它。在 awk 中还可以使用数组。

1．内部变量

awk 的部分预定义变量如表 11-10 所示。

表 11-10　awk 的部分预定义变量

变　　量	说　　明	变　　量	说　　明
ARGC	命令行参数的个数	ARGV	命令行参数数组
ARGIND	命令行参数数组指针	ENVIRON	环境变量数组
ERRNO	文件等操作时出现错误的错误号	FILENAME	awk 浏览的当前文件名
FIELDWIDTHS …	指定域宽，不使用域分隔符	NF	当前文件记录中的域个数
FNR	当前文件的记录数或号	NR	当前记录数或号
FS	域分隔符，默认为白空格	RS	输入时记录分隔符，默认为回车键
OFS	输出域分隔符，默认为空格	ORS	输出记录分隔符，默认为回车键
BEGIN	定义开始块。只在 awk 开始时执行	END	定义结束块。只在 awk 结束前执行

2．用户自定义变量

用户自定义变量的变量名可以由字母、数字和下画线组成，但是不能以数字开头。awk 的变量不用声明类型，awk 可以从变量在表达式中的上下文推导出它的数据类型。如果变量未被初始化，awk 会将字符串变量初始化为空串，将数值变量初始化为 0。必要时，awk 会将字符型变量转换为数值型变量，或者反向转换。

在 awk 中引用自定义变量时不需在其名前面加上标志符“$”。

对变量的赋值语法支持 C 语言格式。

3．域变量

域变量是 awk 中定义用来存储每条记录域值的变量，每条记录的第一个域用$1 表示，第二个域用$2，依次类推。特殊变量$0 用于存储整条记录的内容。

域变量可以像用户自定义的变量一样使用，唯一的区别是它们引用了域。新的域可以通过赋值来创建。域变量引用的域如果没有值，则被赋值为空串。域的值发生变化时，awk 会以内置变量 OFS 的值作为域分隔符重新计算$0 变量的值。

11.3.5　操作符

awk 的操作符可以分为数字型、字符串型、变量型、域和数组元素等。awk 的部分操作符如表 11-11 所示。

表 11-11　awk 的部分操作符

运　算　符	意　　义	运　算　符	意　　义
++,--	自加，自减运算符	?	用于 C 语言表达式:x?y:x
+,-,*,/,%	算术运算符	||,&&,!	或、与、非
=,+=,*=,/=,%=,^=	赋值操作	～,!～	匹配或不匹配
<,<=,==,!=,>=	关系运算	^,**,**=	指数运算符

11.3.6　控制语句

1．awk 的控制语句

（1）if 语句：if（condition）statement [else statement]
（2）while 语句：while（condition）statement
（3）do 语句：do statement while（condition）
（4）for 语句：for（expr1;expr2;expr3）statement
　　　　　　　for（var in arry）statement
（5）控制转移和退出语句：break，continue，exit
（6）语句组：{ statements }

2．语句块

语句块是指用“{ }”括起来的语句组或复合语句，上面的语句组就是其中之一。经常用到的还有 BEGIN 块和 END 块。

（1）BEGIN 块。

一般形式为 BEGIN 后面跟了一个操作块。awk 必须在对输入文件进行任何处理之前先执行 BEGIN 块。实际上，不需要任何输入文件，也能对 BEGIN 块进行测试，因为 awk 要在执行完 BEGIN 操作块后才开始读取输入。BEGIN 操作常常被用来修改内置变量（比如 OFS、RS、FS 等）的值，或为用户自定义变量赋初值。

（2）END 块。

一般形式为 END 后面跟一个操作块。awk 在处理完所有输入行之后才处理 END 块。

11.3.7　常用函数

awk 中有很多内部函数，用户也可以定义自己的函数。awk 的部分常用函数如表 11-12 所示。

表 11-12　awk 的部分常用函数

函　　数	功　　能	函　　数	功　　能
gsub(r,s,)	在$0 中搜索字符串 r 并用 s 替代	gsub(r,s,t)	在字符串 t 中搜索字符串 r 并用 s 替换
index(s,t)	查找字符串 s 中 t 的位置	length(s)	返回字符串 s 的长度
match(s,t)	测试字符串 s 中是否包含匹配模式 t 的字符串	print	显示其变量的内容
sub(r,s)	在$0 中用 s 替换第一次出现的 r	substr(s,p)	返回字符串 s 中从 p 开始的子字符串
substr(s,p,n)	返回字符串 s 中从 p 开始的长度为 n 的子串	toupper(s)	返回字符串 s 的大写形式
tolower(s)	返回字符串 s 的小写形式	atan2(x,y)	y/x 的反正切
sin(x)	x 的正弦	cos(x)	x 的余弦
exp(x)	e^x	log(x)	x 的自然对数

（续表）

函　　数	功　　能	函　　数	功　　能
rand()	返回一个 0～1 之间的随机数	int(x)	x 的整数部分
and(x,y)	x 和 y 的与	or(x,y)	x 和 y 的或
xor(x,y)	x 和 y 的异或	compl(x)	x 的补
lshift(x,cnt)	将 x 左移 cnt 位	rshift(x,cnt)	将 x 右移 cnt 位
sqrt(x)	x 的平方根	system(cmd)	执行 shell 命令 cmd

11.3.8　awk 程序的执行

awk 由模式操作和函数语句序列组成，格式为：

```
pattern { action statement }
function func_name ( param_list ) { statements }
```

模式操作包括模式和行为对，其中每个模式都是正则表达式。在 awk 执行时首先从命令行或程序文件中读取程序，然后解释并执行之。在执行过程中，awk 读取输入文件或标准输入，按顺序检查其中每一个记录是否能进行模式匹配，若能则执行行为操作。

awk 把输入文件的数据读入内存，然后操作内存中的输入数据副本，awk 不会修改输入文件的内容。awk 总是输出到标准输出，如果想让 awk 输出到文件，可以使用输出重定向。

awk 对脚本程序处理的过程如下：

（1）如果 BEGIN 块存在，先执行 BEGIN 块指定的操作。

（2）awk 从输入文件中读取一行，称为一条输入记录（若文件省略，将从标准输入读）。

（3）awk 将读入的记录分隔成域，将第 1 个域放入变量$1 中，第 2 个域放入$2，依次类推。$0 表示整条记录。域分隔符由 awk 内置变量 FS 的值来指定。

（4）把当前输入记录依次与每一个 awk 命令中的 pattern 比较，看是否匹配，如果相匹配，就执行对应的操作。如果不匹配，就跳过对应的操作，直到比较完所有的 awk 命令。

（5）当一条输入记录比较了所有的 awk 命令后，awk 读取输入的下一行，继续重复步骤（3）和（4），这个过程一直持续，直到 awk 读取到文件尾。

（6）当 awk 读完所有的输入行后，如果存在 END，就执行 END 块。

11.3.9　awk 使用示例

（1）显示系统内 root 用户进程的进程号和进程名。

```
#ps -ef | awk '/root/{ print $1, $2, $8 }'
```

（2）显示系统中所有非 root 用户进程，它们是谁的，PID 是多少，在干什么。

```
#ps -ef | awk '!/root/{ print $1, $2, $8 }'
```

（3）计算文件 infile 内数字的和。假设 infile 内容为数字串，每行可有任意多个数字串。

```
#awk '{ for (i=1; i<=NF; i++) s +=$i } END { print s }' ifile
```

（4）显示系统内已经注册（创建）的所有用户名（/etc/passwd 内的所有用户）。

```
#awk -F: '{ print $1 }' /etc/passwd        #-F:指定域分隔符为":"
```

（5）为文本文件增添等号。

```
#awk '{print NR, $0}' ifile
```

（6）对文本文件行中具有某特殊字符串的行执行 shell 命令。

```
#比如，当遇到有 date 的行执行 date 命令。
#awk '/date/{system("date +%D%t%T")}' ifile
```

（7）实现 wc 的功能：统计文本文件中的行数、字数和字符数。

实现 wc 的基本功能，比如 wc file，可由以下代码来完成：

```
#!/usr/bin/awk -f
BEGIN { w=0; l=0; c=0 }
{
w += NF; l++; c += length($0) + 1
}
END { print l, w, c }
```

由于内容较多，代码较长，写在命令行有所不便，故可将其保存到文件，比如 wc.awk，此时执行方法为：

```
#awk -f wc.awk infile
```

在为 wc.awk 增加执行权后，就可以直接执行了，方法是：

```
#chmod +x wc.awk
#./wc.awk infile
```

11.4　Bourne shell 及其编程

Bourne shell 是 Stephen Bourne 在 Bell 实验室设计的一种 shell，简称 bshell（bsh），当时附属于 AT&T。从那时起，许多厂商根据不同硬件平台设计出了不同版本的 UNIX，但在这些不同 UNIX 版本中，bshell 却一直存在，并保持基本一致。bshell 几乎是所有 UNIX 版本的默认 shell。

bashell（bash）是 bshell 的另一个版本，Linux 系统的默认 shell 是 bashell。由于 bashell 兼容 bshell，在本书中以后不再严格区分 bshell 和 bashell，shell 程序设计时也是如此。

11.4.1　特殊字符

shell 使用的特殊字符如下：

（1）*，？，[-]，[!-]：通配符。

（2）<，<<，>，>>，|：I/O 重定向符。

（3）;：命令分隔符。

（4）&：命令后台执行。
（5）"，'，`：三种引号。
（6）\：转义字符。
（7）$：变量引用前导符。
（8）{，}，[，]，(，)：括号。
（9）～：波浪号扩展（为用户的家目录），与$HOME 相同。
（10）&&，||：命令的与或结构及条件执行。

11.4.2　I/O 重定向

1．标准 I/O 的重定向

（1）<：用于输入的重定向；
（2）<<：即时文档（见后述）；
（3）>、2>、&>：分别用于覆盖方式标准输出、标准错误和两者同时重定向；
（4）>>、2>>、&>>：分别用于追加方式标准输出、标准错误和两者同时重定向。

2．标准流间的相互重定向

标准输出和标准错误间的重定向实现方法为：

```
x>&y
```

x，y 的值分别为 1 或 2，代表标准输出和标准错误。shell 对 I/O 重定向的处理是按从左到右的顺序进行的，因此顺序非常重要。例如：

```
#ls > dirlist 2>&1          #同时重定向标准错误和标准输出到文件 dirlist。等同于 ls &> dirlist
#ls 2>&1 >dirlist           #只重定向标准输出，因为标准错误重定向发生在标准输出重定向前
```

3．here 文件或即时文档

另一种输入重定向是即时文档，也叫即时文件，用于在编程时对命令所需的数据进行即时输入。

格式是：

```
cmd <<[-]BEGIN_STR
    here-document
END_STR
```

对于 here 文档的使用有以下说明：

（1）shell 对 BEGIN_STR 不进行扩展（即不进行变量或命令等替换），但可以用引号将其括起来。也就是说，无论 BEGIN_STR 的构成如何，都将被认为是常量字符串或是由引号括起来的常量字符串。

（2）END_STR 必须是一个不含任何引号的被视作常量的字符串，且不能有尾部白空格。

（3）若 BEGIN_STR 被引号括住，则引号被去掉后的结果必须与 END_STR 相同，此时 BEGIN_STR 与 END_STR 之间的内容不进行扩展。若 BEGIN_STR 没有被引号括住，则它必

须与 END_STR 相同，它们之间的内容会进行变量、参数和命令等各种替换。

（4）若使用了-作为 BEGIN_STR 的引导符，则即时文档内容每行的前导白空格都将被自动去除。若不使用-字符，则在 END_STR 的前后都不能有白空格。

因此，即时文档最常用的形式为：

```
cmd <<[-]STR                         cmd <<[-]"STR"
        here-document        或              here-document
STR                                  STR
```

即时文档使用示例如下。

```
cat << !
        -----Here-Document Demo-----
        I am "`whoami`", my Home dir is $HOME, and who are you? please tell me.
!
```

这里的 BEGIN_STR 与 END_STR 均为“！”，且没有使用“-”，故其中的`whoami`和$HOME 都将被替换为操作者的用户名和家目录。如果操作者的用户名为 test，则上述命令的输出为：

```
-----Here-Document Demo-----
I am "test", my Home dir is /home/test, and who are you? please tell me.
```

若要对即时文档命令的标准输出进行重定向，应将命令书写为如下形式：

```
cmd <<[-]STR    > outfile                cmd <<[-]"STR"    > outfile
        here-document            或              here-document
STR                                      STR
```

11.4.3　变量与参数

1．位置参数

当一个 shell 程序被调用时，shell 隐含地为它建立一系列位置参数。这些参数按它们排列的位置被编号为 0，1，…，其中 0#参数为命令本身的名字，1#，2#，…等分别为它所携带的第一个参数，第二个参数，…

例如，在命令

```
myprog    first    xyz    123
```

中，0#参数为程序名 myprog，1#参数为 first，2#参数为 xyz，3#参数为 123。

Shell 脚本程序的位置变量依次表示为$1，$2，…，而脚本程序名表示为$0。就上例而言，$0=myprog，$1=first，$2=xyz，$3=123，…

位置参数可以很多，但能直接访问的只有$0～$9 10 个。对第 10 个以后位置参数，需要使用${}的办法来处理，比如第 11 个位置参数为${11}。当命令行参数超过 10 个时，访问不便，可使用 shift 进行调整，将后边的前移。

2．特殊变量

shell 中有脚本，一些具有特殊意义的预定义变量如表 11-13 所示。

表 11-13　shell 中一些具有特殊意义的变量

变 量	意　　义	变 量	意　　义
$#	传递给 shell 程序的变量个数	$?	上一条命令执行返回值
$@	对于"$@",为 shell 程序的一个个命令行参数；对于$@，为一个个的单词	$*	对于"$*"，为所有命令行参数由分隔符$IFS 连接的单个字符串；对于$*为一个个单词
$n	第 n 个位置参数，n 为小于$#的非负整数	$-	由 set 设置的参数串
$$	当前进程的进程号	$!	最后一次后台执行进程的 pid

说明：$@与$*的区别主要表现在对"$@"与"$*"的处理上，如果不考虑双引号问题，两者是相同的。

3．变量的条件替换

由于 shell 是解释语言，变量不需要声明，也不需要明显的初始化，但应该为变量的初次使用提供某种控制机制，于是就有了变量的条件替换。变量条件替换的形式如下所述。

（1）${var:-val}：当变量 var 未被赋值时而访问时，取默认值 val。

（2）${var:=val}：当变量 var 未被赋值时而访问时，取默认值 val，并把 val 赋给 var。

（3）${var:+val}：当 var 已赋值时，才取 val 的值。

（4）${var:?message}：当变量已设置且不为空则正常替换，则否将 message 送到标准输出后退出 shell 程序。

11.4.4　shell 的状态

在任意时刻，shell 都处于某种状态。这些状态包括各种参数、变量、执行方式和当前工作目录等，可以使用 set 命令来改变或显示 shell 的工作状态。

1．工作目录及改变

shell 的工作目录可用 cd 命令来改变或设置，但在 shell 程序（包括 C 等语言）中，cd 命令改变的目录只对当前进程及其子 shell 进程中有效。当子程序退出后，它改变的工作目录将被自动恢复到原来的位置，或者说，当前进程的父进程并不知道目录的改变。

在 UNIX/Linux 系统内出现了符号链接，若链接的是一个目录，当用户通过符号链接切换工作目录时，就存在逻辑路径和绝对路径的问题，通过参数-L 和-P 可以使用逻辑或物理位置。例如，/usr/tmp 是/var/tmp 的一个符号链接，当用户进入/usr/tmp 时就有绝对路径和相对路径的问题。例如：

```
#cd /usr/tmp            #进入目录/usr/tmp
#pwd -L                 #显示结果为/usr/tmp
#pwd -P                 #显示结果为/var/tmp
```

2．set 与 shell 状态

还有一些与 shell 程序执行相关的状态，它们可以通过 set 命令来设置。set 的用法为：

```
set [--abefhkmnptuvxBCEHPT] [-o option-name] [arg ...]
set [+abefhkmnptuvxBCEHPT] [+o option-name] [arg ...]
```

set 的部分常用参数如表 11-14 所示。

表 11-14　set 的部分常用参数

参　　数	意　　义
-a	设置自动定义环境变量标志，之后定义或修改的变量或函数将作为环境变量
-f	取消路径通配符扩展
-n	只读取 shell 程序的命令做语法等检查而不真正执行，可用于测试 shell 程序是否正确
-v	执行过程中显示读入的行，可用于 shell 程序的跟踪与调试
-x	shell 程序执行过程中，在命令被扩展后、执行前显示该命令及其参数的值，可用于 shell 程序的跟踪与调试
--	如果 args 不存在则取消所有环境变量，否则将 args 设置为新的环境变量
+<参数>	取消以-<参数>方式设置的参数

11.4.5　shell 的调用与变量传递

1．调用 shell 程序

调用 shell 程序，或从一个当前 shell 程序调用另一个 shell 程序时，可让被调用者在子 shell 或当前 shell 中执行，默认是在子 shell 执行。方法如下所述。

（1）在子 shell 执行：

```
shell_prog    args
```

（2）在当前 shell 执行：

```
. shell_prog    args
```

上述方法中调用的子 shell 程序是具有可执行权的，对于无执行权限且具有可执行内容的 shell 程序，可用下面方法来实现：

```
sh shell_prog args
```

2．向子 shell 程序传递变量

当前 shell 的环境变量会自动传递给自己的子 shell 程序，但是自己定义的变量却不能自动传递到下一级，除非子 shell 程序是在当前 shell 执行的。若让在当前 shell 或进程定义的变量传递到下一级，就必须使用 export（参见 3.2.8 节）。

11.4.6 shell 程序设计

1．运算符

传统 bshell 的算术运行符有+、−、*、/、%，更多的其他运行符参见 test 和 expr。

2．bashell 的常用命令

（1）:：空命令。什么都不做，只有返回一个值 0。
（2）. shell_prog：在当前 shell 中执行程序。
（3）;：命令分隔符。同一行中可写多个命令，但命令与命令之间要用“;”分隔。
（4）cd [-L | -P] [arg]：改变工作目录。
（5）pwd [-L | -P]：显示当前工作目录。
（6）echo [args]：显示字符串或变量的值。
（7）exit [n]：强迫 shell 退出[退出码为 n]。
（8）return [n]：函数内强制返回[返回码为 n]。
（9）read [arg1 …]：从标准输入读入变量值（参见 3.2.8 节）。
（10）readonly [arg1 …]：将变量 arg1 等标识为只读，只读变量的值不能再被修改。
（11）export：向子 shell 传递自定义环境变量。
（12）break [n]：从 for、while 或 until 循环内跳出，n 代表跳出的层数。
（13）continue [n]：从 for、while 或 until 循环的开始处执行，n 代表循环的层数。
（14）trap [-lp] [func] signals：设置信号或软中断的处理办法。
（15）times cmd [args]：显示 shell 和 shell cmd 在系统和用户态执行的累计时间。
time cmd [args]：显示 cmd 在系统、用户态的执行时间和累计时间。
（16）umask [-S] [mask]：设置文件创建掩码。

（17）wait [n]：等待某个进程的结束，并报告它的结束状态。若不指定 n，则所有子进程都将被等待。如果 n 为某个不存在的进程，则 wait 返回 127，其他情况下，wait 的返回值与被等待进程的返回值相同。

（18）expr：用于对表达式进行运算。

expr 的功能是计算表达式的值，并将结果显示到标准输出上。当 expr 操作的表达式既不为空也不为 0 时，返回 0；为空或 0 时返回 1；无效语句返回 2，其他错误返回 3。

expr 要求每个参数之间必须用空格分隔，且一些操作符需要使用“\”进行转义。表 11-15 列出了 expr 命令的运算符。

表 11-15　expr 命令的运算符

类　型	运　算　符	功　能
算术运算	+、-	加、减运算符
	*、/、%	乘、除、求余运算符
逻辑运算	ARG1|ARG2	若 ARG1 非空亦非 0 则返回 ARG1，否则返回 ARG2
	ARG1&ARG2	若 ARG1 和 ARG2 均非空亦非 0 则返回 ARG1，否则返回 0
	< <= = == != >= >	比较运算符。真时返回 1，否则返回 0，　=与==意义相同

（续表）

类　型	运　算　符	功　　能
字符串	string : regex	在 string 中匹配正则表达式 regex
	match string regex	在 string 中匹配正则表达式 regex*
	substr string pos length	在 string 中从 pos 开始取最多 length 个字符*
	index string chars	在 string 中查找 char 的开始位置*
	length string	求 string 的长度*
	+ token	将 token 作为普通字符串，其中可能含有关键字或运算符*

注：表中标有*的项目在早期的版本中不支持。

expr 示例如下：

① 计算表达式 " 50 * (40 - 25) " 的值。

```
expr 50 \* \( 40 - 25 \)                    #输出 750
```

② 更新某个变量的值：

```
num=1; num=`expr $num + 5`                 #num 的值 6
```

③ 模式匹配：

```
expr abc : 'a\(.\)c'                        #输出 b
expr abc : '\(a.c\)'                        #输出 abc
expr index    "Who is Mr Shao"    Mr        #输出 8
expr index    "Who is Mr Shao"    Shao      #输出 2
```

（19）eval [arg …]：从它的参数读内容，并将它们传送到 shell 执行，而 eval 再把 shell 的输出结果作为命令进行执行。例如：

```
#cmd=ls; eval `echo $cmd`                    #执行 ls
#eval `id | awk '/uid=0/ { print "isroot=T" }'`   #动态定义变量 isroot
#eval echo \$$#                              #输出最后一个命令行参数（若此为 shell 脚本的一句）
#eval echo \$$(($#-1))                       #输出倒数第二个命令行参数（若引为 shell 脚本的一句）
```

（20）exec：在当前 shell 中执行命令，当前 shell 将被新的执行程序替换或覆盖。exec 的用法为：

```
exec [-lc] [-a name ] cmd args
```

若没有指定选项则 exec 在当前 shell 内执行命令 cmd args；若设定-l 则 shell 在向 cmd 传递第 0 号参数时，在它的前面加上一个“−”号。若指定-a name，则 shell 在向 cmd 传递环境变量时将把 name 作为 cmd 命令的第 0 号参数。若指定-c，则 shell 只是向 cmd 传递一个空环境。例如：

```
#设有一个可能执行程序 dispargs 专用来显示自己的全部命令行参数
#exec -l dispargs a1 b2 c3                  #输出为：-dispargs a1 b2 c3
#exec -a xyx dispargs a1 b2 c3              #输出为：xyz a1 b2 c3
```

（21）shift：调整位置参数，其用法为：

shift [n]

shift [n]的功能为将第 n+1，n+2，…变量的值变为$1，$2，…n 必须为大于 0 的整数，默认值为 1。若 n 的值等于 0 或大于$#，命令执行失败，返回非零值。若执行成功则$#= $#− n。

（22）test：用于测试条件表达式的值。test 命令主要用于测试某条件的返回值，成功或真时为 0，不成功或假时为非 0，其用法为：

test condition

test 可用于以下 4 种条件的测试：两字符串间的关系；两整数间的关系；文件是否存在或是否具有某种状态或属性；以上几种条件通过“与”或“或”的组合。

① 用于字符串间关系测试如表 11-16 所示。

表 11-16　字符串间关系测试

运　算　符	意　　义	示　　例
str1 = str2 或 str1 == str2	相等时为真	x=abc;y=acd;test $x = $y 为假
str1 != str2	不等时为真	x=abc;y=acd;test $x != $y 为真
str	非空时为真	x=abc;test [$x]为真
-n str	长度大于 0 时为真	x=abc;test -n $x 为真
-z str	长度等于 0 时为真	x=abc;test -z $x 为真
str1 < str2	排序后 str1 在 str2 前为真	x=abc;y=acd;test $x < $y 为真
str1 > str2	排序后 str1 在 str2 后为真	x=abc;y=acd;test $x > $y 为假

② 用于两整数间关系的测试如表 11-17 所示。

表 11-17　两整数间的比较运算符

运　算　符	意　　义	示　　例
-eq	两数相等时为真	x=1;y=2;$x -eq $y 为假
-ne	两数不等时为真	x=1;y=2;$x -ne $y 为真
-gt	前数大于后数时为真	x=1;y=2;$x -gt $y 为假
-ge	前数大于或等于后数时为真	x=1;y=2;$x -eq $y 为假
-lt	前数小于后数时为真	x=1;y=2;$x -lt $y 为真
-le	前数小于或等于后数时为真	x=1;y=2;$x -le $y 为真

比较示例如下：

```
#str1=123;str2="123"
#test $str1 = $str2; echo $?            #结果为 0
#test $str1 -eq $str2; echo $?          #结果为 0
```

③ 有关文件的测试如表 11-18 所示。

表 11-18　有关文件的测试表达式

测 试 符	意 义	测 试 符	意 义
-r file	可读时为真	-w file	可写时为真
-x file	可执行时为真	-f file	为正则文件时为真
-c file	字符设备文件时为真	-b file	块设备文件时为真
-d file	目录文件时为真	-e file	文件存在时为真
-h/-L	符号链接时为真	-H file	Semaphore 时为真
-u file	设置 suid 时为真	-g file	设置 sgid 时为真
-k file	设置 sticky 时为真	-p file	命名 pipe 时为真
-s file	长度>0 时为真	-M file	共享内存文件时为真
f1 -nt f2	文件 f1 比 f2 新时为真	f1 -ot f2	文件 f1 比 f2 旧时为真
f1 -ef f2	文件 f1 与 f2 在同一设备上且 i 节点号相同时为真		

④ 复杂条件测试。当测试语句中的条件不止一个时，可根据实际情况使用逻辑与(-a)、或(-o)、非(!)进行组合。例如：

```
#test -s file -a -r file              #如果文件长度>0 且可读时为真
#test $val -lt 0 -o $val -gt 7        #如果变量 val 的值<0 或>7 时为真
```

（23）[]：另一种执行 test 命令的方法

条件判断是经常用到的，因此 test 语句在 shell 程序设计时是非常有用的。但是，采用这种方法操作显得有些不便，于是又引入了另一种既方便又直观的办法。具体做法是把判断条件语句放在一对方括号中，来替代 test 命令。若表达式中用于测试的条件不止一个，则两个用于测试的式子间要用另一种逻辑运算符来组合，它们是与（&&）、或（||）和非（!）。例如，上面的 test 测试命令，可以改为：

```
#[ -s file ] && [ -r file ]           #如果文件长度>0 且可读时为真
#[ $val -lt 0 ] || [ $val -gt 7 ]     #如果变量 val 的值<0 或>7 时为真
```

（24）dirname、basename：确定文件的路径名和基本名

dirname 和 basename 用于确定一个带有路径文件名的路径名和文件名，其用法为：

```
dirname    names
basename [-a |-s, --suffix=suffix ]    names
```

常用参数和选项有：names 为带有路径的文件名；-a 用于多文件名处理，若不使用-a 则只处理第一个；-s，--suffix=suffix 用于在输出中删除扩展名。

dirname 用于输出带有路径文件名的路径名，若没有路径则输出“.”，表示当前路径。basename 用于输出带有路径文件名的纯文件名，若使用了-s suffix 或 --suffix=suffix，则只输出文件的主名。示例如下：

```
#dirname /usr/bin/ /usr/include/stdio.h              #输出/usr 和/usr/include
#basename -a /usr/bin/ /usr/include/stdio.h          #输出 bin 和 stdio.h
#basename -a -s .h /usr/bin/ /usr/include/stdio.h    #输出 bin 和 stdio
```

3．bash 的其他扩展

1）参数或变量条件扩展

除了 bsh 的${var:-val}、${var:=val}、${var:+val}和${var:?val}外，bash 又增加了下以扩展方式：

${#var}：变量 var 值的长度。

${var:off}、${var:off:len}：var 中从 off 开始长度为 len 的子串。

${var#val}、${var##val}：从 var 头部去除由模式 val 匹配的最小和最大部分，返回剩余部分。

${var%val}、${var%%val}：从 var 尾部去除由模式 val 匹配的最小和最大部分，返回剩余部分。

${var/pattern/str}、${var//pattern/str}：pattern 为像文件扩展一样被扩展的匹配模式，var 也被扩展，且将 pattern 在 var 中的最大匹配替换为 str。前者仅进行首次匹配替换，后者将进行所有的匹配替换。如果 pattern 的首字符为#/%，则从行首/尾进行匹配和替换。当 str 为 NULL 时，删除匹配的串。如果 var 为*或@，则替换将在位置参数上依次进行。

2）“{ }”扩展与路径名扩展

bash 还支持“{}”扩展，比如 a{a, b, c,}d 将被替换为 aad,abd,acd，这种替换常用在命令行参数替换中。比如

```
mkdir /home/zh3/a{1,2,3}dir
```

将在/home/zh3/创建 a1ddir、a2dir 和 a3dir 目录。

3）$'str'和$"str"与字符串扩展

形如$'str'结构将会被扩展成一个字符串，其中的转义字符会按照 ANSI C 的标准被替换。

```
c=$'\a'                    #c 为响铃，而非字符串'\a'
str=$'\n\r'                #str 为回车换行，而非字符串'\n\r'
```

形如$"str"结构会使字符串 str 根据当前 locale 来翻译。事实上，这种替换就可以理解为双引号内的替换。

4）“(( ))”和“[[ ]]”

“((expr))”对算术表达式 expr 求值，如果这个值不是零，则返回状态是零，否则返回 1。比如：

```
x=5; ((y=x++)); echo "x=$x    y=$y"          #x=6 y=5
```

“[[expr]]”对条件表达式 expr 求值，并据其结果返回 0 或者 1。在“[[”和“]]”中间的内容不进行文件名扩展，但却可进行“~”扩展、参数和变量扩展、算术扩展、命令替换、进程替换。如果使用了“==”或“!=”，则其右边会被看成一个模式，并且按照模式匹配规则进行匹配。

5）$((expression))：计算表达式 expression 的值

这是一个比 expr 计算能力更强的工具，在 expression 中可以包括 C 语言除指针以外的几乎所有运算符。例如：

```
#x=3;y=4;z=5;x=$((z**2-x*y));echo $x                 #结果为 13
```

```
#echo $((z**2-(x**2+y**2)))                    #结果为 0
```

6）$(cmds)：命令替换的另一种形式

```
$(cmds)是命令替换的另一种形式，且比 cmds 具有更好的表现。
```

7）进程替换

进程替换的一般形式为：

```
>(command)
<(command)
```

前者用于向命令 command 提供输出，后者用于从 command 得到输入。说明：在"<"或">"与圆括号“(”之间不能有空格。

进程替换经常出现在文件名作为参数的地方。

比如：

```
#diff <(ls -a /home/zh3) <(ls -a /home/li4)     #比较/home/zh3 与/home/li4 内容的差别
#tar cvf >(bzip2 -c > /tmp/UNIX.tbz2) ~/UNIX     #将~/UNIX 打包备份为/tmp/UNIX.tbz2，等同于
#tar cvfj /tmp/UNIX.tbz2 ~/UNIX
```

4．函数

在 shell 编程时可以使用函数，但在使用之前函数必须是先定义好的，且必须出现在调用语句的前面，函数支持递归调用。

函数定义时不必使用形式参数，对函数参数的引用如同在 shell 中使用位置参数一样。在函数体内的$1 代表函数的第一个参数，$i 代表函数的第 i 个参数,$#代表函数参数的个数，$*代表所有参数，$0 不代表函数名，而是它所在的 shell 程序名。函数定义方法为：

```
func_name( ) {
        函数体
}
```

函数调用方法为：

```
func_name param1 param2 …
```

函数被定义以后可以作为命令来使用。现在我们定义函数 sysinfo()，如下所示。

```
#!/bin/sh
sysinfo() {
    echo "$0 has $#param(s),they are: $*"
    echo "I have $#param(s), and the 1'st param is: $1"
    date +"%D%t%T"
    uname –s
}
```

可以像使用 shell 命令一样使用函数，方法是：

```
sysinfo x1 x2 x3               #函数的使用。观察参数的使用，并注意输出中的$0
sysinfo x1 x2 x3 | wc          #函数与 shell 命令的管道链接
```

5．控制结构

1）if-then-fi 结构

if-then-fi 结构的语法为：

```
if test_cmd
then
        cmd_1
        …
        cmd_n
fi
```

在此结构中，if 和 then 是两个语句，若写在同一行则必须用“;”分隔。可以将上述结构写为：

```
if test_cmd; then cmd_1; …; cmd_n; fi
```

if-then-fi 结构的示例如下。

设计一个程序，带一个（目录）参数，实现功能：

（1）首先判断是否带有一个参数，若无，则显示用法信息后报错返回 1；

（2）若带有参数，则首先判断该参数作为文件是否存在，若不存在，报错返回 2；

（3）若存在，判断是不是目录，若是则列目录的内容后返回 0；否则，提示用户参数不是目录，显示提示信息后返回 3。

设脚本程序名为 sh01.sh，其内容如下所示。

```
#!/bin/sh
#让系统调用 bshell 解释程序，此行不能直接跟在#!/bin/sh 的后面
if [ $#-lt 1 ]                      #判断是否有参数
then                                #无参数
    echo -e "Usage:\n$0 dir"        #显示用法信息
    exit 1                          #返回 1
fi
if [ ! -e "$1" ]; then              #判断是否存在，若不存在返回 2
    echo -e "$1 doesn't exist!"; exit 2
fi
if [ -d "$1" ]; then                #判断是否为目录
    ls $1/*; exit 0;                #列目录内容，返回 0
fi
echo " $1 is not a directory"       #警告信息
exit 3                              #返回 3
```

2）if-then-elif-else-fi 结构

```
if-then-elif-else-fi 结构为：
        if test_cmd
        then
            statements
```

```
elif test_cmd
then
        statements
fi
```

采用此结构，sh01.sh 可改写为如下所示的 sh02.sh：

```
#!/bin/sh
if [ $#-ne 1 ]; then echo -e "Usage:\n$0 dir"; exit 1
elif [ ! -e "$1" ]; then echo -e "$1 doesn't exist!"; exit 2
elif [ -d "$1" ]; then ls $1/*; exit 0
else echo "Warn: $1 is not a directory";exit 3
fi
```

说明：if-fi 结构可以嵌套多层。

3）case 结构

case 结构的格式为：

```
case var in
val1)
        cmd11; …; cmd1n1;;
val2)
        cmd21; …; cmd2n2;;
        … … … … … …
*)
        default_action;;
esac
```

注意 esac 为 case 的倒序书写。

4）while 结构

while 结构的格式如下：

```
while condition
do
        do_list             #命令或语句系列
done
```

while 和 case 结构使用示例如下。

设计一个菜单程序，循环接收键盘输入，根据所输入的键来执行不同的功能，直到接收到 0、q 或 Q 键退出。要求捕获信号 2。

为了简化设计又能说明问题，这里提供 3 个功能：func1、func2 和 Quit。实现 func 1 和 func 2 的功能键分别“1”和“2”，对应函数分别为 func1()和 func2()；用于退出的功能键为“0”、“q”或“Q”。设程序名为 sh03.sh，其代码如下所示。

```
#!/bin/sh
func1( ){                                  #定义菜单功能函数 func1，用于完成功能 1
                echo "This is function1!"
```

```
}
func2( ){                                    #定义菜单功能函数 func2，用于完成功能 2
        echo "This is function2!"
}
trap "" 2                                    #捕获信号 2，忽略^C
#主程序
while true                                   #循环接收用户输入
do
    clear                                    #清屏
    echo -n "
+------------------------------------------------------------+
|     1. func1       2. func2      0/q/Q.Exit              |
+------------------------------------------------------------+
          Please get a choice: "
    read x
    case $x in
        1)                                   #处理键 1，执行 func1
            func1;;
        2) func2;;                           #处理键 2，执行 func2
        0|Q|q) exit $?;;                     #处理 q 或 Q：返回上条命令执行结果
        *) echo -e "\a";;                    #响铃表示非法输入
    esac
    read -p "Press Enter to continue: " x    #供观察用，回车继续下一轮循环
done
```

sh03.sh 只接收 0、1、2、q 和 Q，输入其他键时响铃报警。

5）until 结构

until 结构如下：

```
until condition
do
    do_list                 #命令或语句系列
done
```

until 结构编程示例如下。

设计一个程序，计算前 10 个正整数的和。设程序名为 sh04.sh，代码如下所示。

```
#!/bin/sh
x=0; y=1                          #x 和 y 分别为和和控制变量，分别被初始化为 0 和 1
until [ $y -gt 10 ]
do
    x=`expr $x + $y`              #x=x+y
    y=`expr $y + 1`               #y=y+1
done
echo "1+ ... +10=$x"              #输出结果
```

6）for 结构

for 结构如下：

```
for var in val1 val2 …
do
        do_list
done
```

编程示例如下。

设计一个程序，从命令行接收一些数值参数并计算这些参数的和。设程序名为 sh05.sh，代码如下所示。

```
#!/bin/sh
x=0                           #x 为和变量，初始化为 0
for y in $*                   #遍历所有命令行参数
do                            #没有判断参数的合法性，因为它们可能是非数值数据
        x=`expr $x + $y`      #累加 x=x+y
done
echo "The Sum is: $x"         #输出结果
```

7）从循环中跳出或继续下一个循环

从循环中跳出：

```
break [n]
```

继续下一个循环：

```
continue [n]
```

循环与 break 编程示例如下。

设计一个程序，计算前 n 个正整数的和，n 由命令行输入。设程序名为 sh07.sh，代码如下所示。

```
#!/bin/sh
x=0; y=1                                  #x 和 y 分别为和和控制变量，分别被初始化为 0 和 1
while true
do
        x=`expr $x + $y`; y=`expr $y + 1`  #x=x+y; y=y+1
        if [ $y -gt $1 ]; then break; fi   #$1 为命令行参数，要为正整数
done
echo "1+...+$1=$x"                        #输出结果
```

8）使用与或结构有条件执行命令

（1）与结构的使用。

与结构是用&&连接的命令序列：

```
cmd1 && cmd2 && cmd3 …
```

在此结构中的语句是按从前到后的顺序执行的，若执行中某个命令执行失败（返回码不为 0），则后面的命令不再执行。

（2）或结构的使用。

或结构是用||连接的命令序列：

```
cmd1 || cmd2 || cmd3 …
```

在此结构中，只有前面的命令执行失败（返回值为 0），后面的命令才会得到执行，但若有一条命令执行成功了，后面的命令就不再执行。

与结构和或结构可以组合使用，但组构太长将导致程序的可阅读性降低。

9）命令分组

shell 中若干命令可以放在一个执行单元内组成命令组执行，组成命令组可用以下两种形式。

(cmds)：在子 shell 中执行。子 shell 执行的结果不影响当前 shell 程序变量的值。

{ cmds; }：在当前 shell 中执行。当前 shell 中执行的结果将影响当前 shell 的变量值。

命令分组简单示例程序为 sh07.sh，其代码如下所示。

```
#!/bin/sh
N=2; printf "Before ( ): N=%d | " $N;
( a=2;b=3;N=`expr $a + $b`; printf "In (): N=$N | " )
printf "After (): N=$N\n"                    #结果为 2
N=2; printf "Before {}: N=%d | " $N;
{ a=2;b=3;N=`expr $a + $b`; printf "In {}: N=$N | "; }
printf "After {}: N=$N\n"                    #结果为 5
```

sh07.sh 的输出结果如下，根据输出结果，可以核实(cmds)与{ cmds; }的区别。

```
Before ( ): N=2 | In (): N=5 | After (): N=2
Before {}: N=2 | In {}: N=5 | After {}: N=5
```

10）递归

shell 支持递归调用。但由于 shell 程序及函数的返回值只有 8 位，故不能希望像 C 语言那样通过函数的返回值来实现递归计算数的阶乘等。

如下所示的脚本程序 sh08.sh 代码为通过递归实现递归列目录的一种方法。

```
ls_r() {
  for f in $1/*; do
     ls -d   $f
     if [ -d $f ]; then
          ls_r $f;                #递归调用自己
     fi
  done
}
if [ $#-eq 0 -o ! -e "$1" ]; then echo -e "Usage:\n$0 dir"; exit 1; fi
if [ -d "$1" ]; then ls_r $1          #列目录及其子目录
else ls $1
fi
```

6．shell 编程综合示例

（1）示例 1 即时文档使用与系统用户查询。

使用即时文档编写一个 shell 程序，要求：

① 在即时文档内部进行参数或命令替换，显示操作者的用户名，并提示操作者输入一个用户名。

② 接收用户从键盘读入一个字符串，首先判断该输入内容是不是系统中已创建的用户，若不是则显示错误“X is not a user”并返回 2；若是，再判断该用户是否已登录到系统中，若已登录则显示“X is logined”，并返回 0；否则输出“X not in system now!”，并返回 1（说明：X 为用户的输入内容）。

设脚本程序名为 sh09.sh，其代码如下所示。

```
#!/bin/sh
cat << !                                                    #开始（首次）使用即时文档
   I am \"`whoami`\", please give me a user name:
!
read x
if [ -z "$x" ]; then echo "Invalid user name"; exit 3; fi   #判断$x 是否为空
grep -qw "$x" << EOF                                        #开始（第 2 次）使用即时文档
`cut -d: -f1 /etc/passwd`                                   #获取所有用户或使用
#`awk -F: '{ print $1 }' /etc/passwd`
EOF
if [ $? -ne 0 ]; then echo "\"$x\" is not a user"; exit 2; fi
v=""
y=`who | awk '{ print $1 }'`                                #正在工作用户
for z in $y;    do
   case $z    in
       $x) v=$x;;
   esac
done
if [ "$x" = "$v" ]; then                                    #正在工作
       echo "\"$x\" is logined!"; exit 0;
else                                                        #未登录
       echo "\"$x\" is not in system now!";exit 1;
fi
```

（2）示例 2 为 chroot 做准备。

设计一个 shell 程序，为 chroot 做准备（参见 6.5.3 节）。具体对于

```
chroot    newroot    [cmds]
```

的执行环境做自动化准备。这里，cmds 为命令（可以一次指定多个），newroot 为 cmds 运行时的新根目录。设实现脚本程序名为 sh10.sh，其代码如下所示。

```
#函数：用于处理每个带有绝对路径的命令
do_copy() {
```

```
    dir=`dirname $2`                          #获得路径
    if [ "$dir" = '.' ]; then return 1; fi #若无路径，非所需文件，返回 1
    mkdir -p $1$dir 2>/dev/null               #在 newroot 下创建对应目录
    cp $2 $1$dir 2>/dev/null                  #将原绝对路径中的文件复制到新建目录
    return $?                                 #返回
}
Usage( ) { echo -e "$1\nUsage:\n $0 NewRoot cmd1    …"; }
#主程序
#检查有无 newroot 指定，若无则退出
if [ $#-eq 0 ]; then Usage "No path assigned!"; exit 1; else path=$1; fi
shift                                         #命令参数前移一位。原第 2 个为需要处理的命令
#检查有无命令，若无则退出，但可有多个
if [ $#-eq 0 ]; then Usage "No cmds assigned!"; exit 2; fi
mkdir -p $path 2>/dev/null                    #创建 newroot
while true; do                                #循环依次处理每一个命令
    if [ $#-eq 0 ]; then break; else cmd=$1; fi    #0 表示完毕
    cmd=`which $cmd`                          #查找命令在原系统中的位置
    if [ $? -ne 0 ];then shift; continue; fi  #若出错，则一下个
    echo $cmd                                 #显示正在处理的命令
    do_copy $path $cmd#处理命令本身
    dep=`ldd $cmd`;                           #获取命令依赖动态链接库
    if [ $? -ne 0 ];then shift; continue; fi  #出错或无，则继续下一个
    for x in $dep;do                          #处理每个依赖的动态链接库
        echo $x|grep -q ':';
        if [ $? -eq 0 ]; then continue; fi    #命令本身(已处理过不再处理)
        do_copy $path $x        #处理依赖的动态链接库(说明：$x 中有非法文件名，由 do_copy 忽略之)
    done
    shift                                     #下一个命令
done
```

（3）示例 3 向系统中有效用户发信息：已登录者发信息，未登录者发邮件。

编写 shell 程序，实现向系统中的有效用户发消息。要求：

① 程序至少要有 1 个参数为要发的信息内容；

② 若用户已经登录，则将信息直接写到用户终端上；否则将信息以邮件形式发给用户，邮件题目为“Hello $uname”。

说明：这里的有效用户指的是系统中已经存在且是可登录者（即非伪用户），主要表现为登录 shell 为 nologin 或 false 等。

设实现此功能脚本程序名为 sh11.sh，其代码如下所示。

```
#!/bin/sh
if [ $#-lt 1 ]; then                          #如果不带有参数，则显示用法后返回 1
    echo -e "Usage:\n$0 Text_to_Send"; exit 1;
fi
#查找所有有效用户名，并存入 users 变量
users=`grep -E -v '(/nologin)|(/false)$' /etc/passwd | awk -F: '{ print $1 }'`
```

```
#echo $users
for uname in $users; do
   if [ "$uname" = `whoami` ]; then continue; fi                                   #不给自己发
   echo "$*" | { write $uname &>/dev/null || mail -s "Hello $uname " $uname; }    #写信息
done
```

脚本程序的关键部分是：

```
echo "$*" | { write $uname &>/dev/null || mail -s "Hello $uname " $uname; }
```

它的功能是将所有命令行参数要么直接写到已经登录用户的终端上，要么以邮件形式发送给用户（需要 sendmail 支持）。由此可见，命令组可以作为一个“命令”或“函数”使用。

需要说明的是，本例的主旨在于说明命令分组和复合命令的用法，就功能而言是不全面的，它不能找到所有不能登录的用户，比如 halt，sync，shutdown 等。还有，当一个用户同时在多个终端上登录时，它不能将信息写到同一用户所使用的每个终端上。

11.4.7　命令行参数与选项的处理

在 shell 脚本程序中时常需要处理命令行参数和选项，本部分将介绍 shell 脚本中对命令行参数和选项的处理。

1．选项与参数

为了处理命令行参数和选项，C 提供了 getopt/getopt_long 等函数（参见在线手册），C++ 的 boost 提供了 Options 库。在 bshell 中，处理命令行参数和选项的命令是 getopts 和 getopt。getopts 和 getopt 功能相似但又不完全相同，其中 getopt 是独立的可执行命令，而 getopts 是 bshell 的内置命令。

2．getopt 命令

getopt 的功能是对一字符串使用指定预期选项标志（选项名）进行分析，并从中分出选项和非选项参数，将选项部分排在非选项字符串的前面，两者之间插入“--”。因此 getopt 常用来解析命令行参数，并从中分解出选项和参数，其用法为：

```
getopt optstr paramstr
```

optstr 是由用作选项的字符构成的串，如果一个选项需要一个参数值，则在其后紧跟一个冒号“:”；paramstr 是由类似命令行参数一样的字符串，有以“-”或“+”或“--”开头的选项及值，也有一般字符构成的参数。比如，对于字符串“-a -b BBB -c s5 c6 -d”,如果选项标志为 a、b、c 和 d，且 b 需要一个参数值，则 optstr 可以写为“ab:cd”，paramstr 若为“-a -b BBB -c s5 c6 -d”，则此时的 getopt 命令可以书写为：

```
getopt ab:cd    -a -b BBB -c s5 c6 -d
```

执行此命令的目的是分析 paramstr 中有没有所需要的选项及其值。因此，如果 paramstr 是某个命令的命令行参数，optstr 是命令所选项，getopt 可从命令行参数中挑选出其中的选项及其参数的值。

在分析时，若遇到 paramstr 中有不以“-”、“+”和“--”开始的项就认为选项部分结束，其后部分为非选项参数，且在选项和非选项参数之间插入“--”作为分隔符。在输出时，每个选项、参数等项之间以空格分隔。

需要说明的是，此命令及后述的 getopts 在不同的 UNIX 系统中存在较大的差异，比如 Linux，不论命令行参数按什么位置排列，总能将 paramstr 中所有选项按顺序排在非选项的前面，而其他早期的 UNIX 都在首次遇到非选项字符就结束工作，而后的内容都认为是非选项字符。因此，在使用不同的 UNIX 系统时，最好将所有选项部分写在参数的前部，避免将选项误认为非选项。

针对选项字符 ab:cd 和命令行参数串-a -bbbb xyz -c 123 -d uvw，可编写如下所示的脚本程序 sh12.sh，用于从命令行参数中分出选项和参数。

```
#!/bin/sh
echo $*                                                 #显示原始命令行参数
set -- `getopt ab:cd $*`                                #以 getopt 的结果重置命令行参数表
if [ $? -ne 0 ]; then exit 1; fi                        #检查 getopt 的执行状态
echo $*                                                 #显示重置后命令行参数表
aFLG=0; bFLG=0; cFLG=0;dFLG=0; bARG=#设置状态值，初始为 0 或空
while [ $1 != -- ]; do                                  #扫描参数表
        case $1 in
                -a) aFLG=1;;                            #-a
                -b) bFLG=1; bARG=$2; shift;;            #-b 及其值
                -c) cFLG=1;;                            #-c
                -d) dFLG=1;;                            #-d
        esac
        shift                                           #下一个参数
done
shift                                                   #跳过--之后为非选项部分
echo $*                                                 #显示剩余的（非选项）参数
echo "aFLG=$aFLG bFLG=$bFLG bARG=$bARG cFLG=$cFLG dFLG=$dFLG" #显示挑选的结果
```

程序首先显示原始位置参数，然后调用 set -- `getopt ab:cd $*`重置位置参数表，在显示新的位置参数表后，使用 while 循环扫描位置参数表。扫描时，每遇到一个选项就将其对应的标志置 1，然后做一次 shift，使后边的参数前移。对于-b（如果存在）不仅将 bFLG 置 1，而且多做了一次 shfit，将其参数的值赋给 bARG。当遇到“--”时结束扫描，最后剩余的部分为非选项参数。在程序的最后显示了各个参数对应的状态值以确定是否发现了某个选项。可按以下方式执行：

若以-a -bbbb xyz -c 123 -d uvw 作为命令行参数运行之

```
#./sh12.sh -a -bbbb xyz -c 123 -d uvw
```

结果为：

```
-a -bbb xyz -c 123 -d uvw                  #原始命令行参数
-a -b bb -c -d -- xyz 123 uvw              #经 set 处理后的位置参数
xyz 123 uvw                                #非选项参数
aFLG=1 bFLG=1 bARG=bb cFLG=1 dFLG=1        #最后结果
```

3．getopts 方式

getopts 是 shell 的内置命令，主要用于分析命令行参数，从中分出选项和非选项部分，其用法为：

```
getopts optstr opt
```

optstr 的意义与 getopt 相同，opt 为临时变量，用于存放刚刚从命令参数取到的选项字符。getopts 使用几个变量：OPTIND、OPTARG 和 OPTERR。OPTIND 为指向命令行参数的位置指针；OPTARG 为选项字符所对应的参数值；OPTERR 用于控制 getopts 对错误的处理方式，当 OPTERR=0 时不显示处理过程中的错误，否则将显示处理过程中遇到的错误。如下所示的 sh13.sh 为使用 getopts 处理命令行参数的简单例子。

```
#!/bin/sh
#OPTERR=0                                   #为 0 时可以屏蔽错误信息输出
aFLG=0; bFLG=0; cFLG=0; dFLG=0; bARG=       #设置状态值，初始为 0 或空
while    getopts 'ab:cd' opt; do            #从命令行参数获取选项
    case $opt in
        a) aFLG=1;;                         #-a
        b) bFLG=1; bARG=$OPTARG;;           #-b 及其值
        c) cFLG=1;;                         #-c
        c) dFLG=1;;                         #-d
        ?) echo "Usage: $0 options";;       #default
    esac
done
N=`expr $OPTIND - 1`; shift $N              #指向剩余的非选项部分（必须这样做）
echo $*                                     #显示剩余的部分
echo "aFLG=$aFLG bFLG=$bFLG bARG=$bARG cFLG=$cFLG dFLG=$dFLG"
```

可按以下方式执行：

```
#./sh13.sh -a -bbbb xyz -c 123 -d uvw
    xyz -c 123 -d uvw
    aFLG=1 bFLG=1 bARG=bbb cFLG=0 dFLG=0
```

在遇到 xyz 这个非选项参数时结束扫描。由此可以看出，同一个系统内容，不同的命令在处理同一问题时表现也不完全相同。问题在于 getopts 是对原始命令行参数串处理的，如果能将 getopt 和 getopts 两者结合起来使用，将会得到更好的效果。

4．命令行参数综合处理示例

任务：设计 bshell 脚本程序，处理命令行参数，结合 awk 实现类似 cat 命令的-E、-n 功能。要求：

（1）-n 为所显示的文本文件添加行号；

（2）-E 指定在行末显示的结束字符，默认无；

（3）-e 抑制文件名的输出；

（4）-c 在执行-n 或-E 后统计文件中的行数；

（5）-C 只统计文件的行数。

设实现此任务脚本程序名为 sh14.sh，其代码如下所示。

```
#!/bin/bash
eFLG=`getopt cCeE:n $*`; EFLG=$?                  #临时借用 EFLG 和 eFLG
if [ $EFLG -ne 0 ]; then echo -e "Usage:\n$0 [-cCeE:n] [filenames]";exit $EFLG; fi
set --   $eFLG
#OPTERR=1      #To disable the Output of error message
EFLG=0; eFLG=0; CFLG=0;cFLG=0; NFLG=0; EARG= #初始化各种标志：置 0 或空
while getopts 'cCeE:n' OPT; do                    #从命令行参数获得选项
        case $OPT in
                c) cFLG=1;;                       #-c
                C) CFLG=1;;                       #-C
                n) NFLG=1;;                       #-n
                e) eFLG=1;;                       #-e
                E) EFLG=1; EARG="$OPTARG";;       #-E
                ?) echo -e "Usage:\n$0 [options] [filenames]";exit 1;;
        esac
done
N=`expr $OPTIND - 1`; shift $N   #清除命令行参数中已经处理的部分，剩余部分为文件名
while [ $#-ge 1 ]; do                             #处理每一个文件
        f=$1                                      #取一个参数作为文件名
        if [ $eFLG -eq 0 ]; then echo $f; fi      #是否显示文件名
        if [ $CFLG -eq 0 ]; then                  #如果没有指定-C
                if [ $NFLG -eq 1 ]; then          #如果显示行号
                        if [ -n $EARG ]; then     #如果显示行末结束符
                                awk "{ print FNR, \$0 \"$EARG\" }" $f
                        else                      #不显示行末结束符
                                awk '{ print FNR, $0 }' $f
                        fi
                else                              #如果不显示行号
                        if [ -n $EARG ]; then     #如果显示行末结束符
                                awk -v E=$EARG '{ print $0, E }' $f
                        else                      #不显示行末结束符
                                awk '{ print $0 }' $f
                        fi
                fi
        fi
        if [ $cFLG -eq 1 -o $CFLG -eq 1 ]; then   #-c 或-C
                awk 'END {print FNR}' $f
        fi
        shift
done
```

在此程序对 awk 的使用中，采用了不同的方式，这里涉及如何向 awk 脚本传递 shell 变

量的问题。首次调用 awk 时使用的是双引号，意在进行参数替换，传递 shell 变量 EARG 到 awk，但 awk 脚本的内容是{ print FNR \$0 \"$EARG\" }；在第三次调用时，也是向 awk 传递 shell 变量，但这里采用的是单引号，理由是因为使用了选项-v E=$EARG 定义了 awk 的变量 E。sh13.sh 执行示例如下：

```
#./sh14.sh    -n sh13.sh              #为 sh13.sh 加行号
#./sh14.sh    -n -E'$' sh13.sh        #加行号且在每行末显示$
#./sh14.sh    -nc -E'$' sh13.sh       #加行号且在每行末显示$,最后显示总行数
#./sh14.sh    -c sh13.sh              #仅显示 sh13.sh 的内容，且在最后显示总行数
#./sh14.sh    -C sh13.sh              #仅仅显示 sh13.sh 的总行数
```

11.4.8　shell 程序调试

shell 程序的调试是通过在程序加入相关语句，让 shell 程序在执行过程中显示一些可供参考的“调试信息”。一般调试方法是通过 set 命令设置用于调试的参数-n、-v 和-x 等。当然用户也可以在 shell 程序中的适当位置加入一些 echo 或 printf 命令用于调试、跟踪或观察。

设当前目录下有子目录 yy，且 yy 内有文件 y.c、yl 和 y.o，在 if-fi 结构示例程序 sh01.sh 代码中分别加入 set -v 或 set -x，并将其保存为 sh15.sh 或 sh16.sh，按方式：

```
#./sh15.sh yy
#./sh16.sh yy
```

Sh15.sh 源代码如下：

```
#!/bin/sh
#让系统调用 bshell 解释程序，此行不能直接跟在#!/bin/sh 的后面
set -v      #或  set -x                #在这里添加 set -v  或 set -x
if [ $#-lt 1 ]                         #判断是否有参数
then
    echo -e "Usage:\n$0 dir"           #显示用法信息
    exit 1                             #返回码为 1
fi
if [ ! -e "$1" ]; then                 #判断是否存在，若不存在返回 2
    echo -e "$1 doesn't exist!"; exit 2
fi
if [ -d "$1" ]; then                   #判断是否为目录
    ls $1/*; exit 0;                   #列目录内容，返回 0
fi
echo " $1 is not a directory"          #警告信息
exit 3                                 #返回码为 3
```

加入 set -v 的执行调试信息输出如下：

```
if [ $#-lt 1 ]                         #判断是否有参数
then
    echo -e "Usage:\n$0 dir"           #显示用法信息
    exit 1                             #返回码为 1
```

```
fi
if [ ! -e "$1" ]; then                  #判断是否存在，若不存在返回 2
    echo -e "$1 doesn't exist!"; exit 2
fi
if [ -d "$1" ]; then                    #判断是否为目录
    ls $1/*; exit 0;                    #列目录内容，返回 0
fi
yy/y.c      yy/yl   yy/y.o
```

加入 set -x 的执行调试信息输出如下：

```
+ '[' 1 -lt 1 ']'
+ '[' '!' -e yy ']'
+ '[' -d yy ']'
+ ls yy/y.c yy/yl yy/y.o
yy/y.c      yy/yl   yy/y.o
+ exit 0
```

根据调试信息的情况可以观察程序的每一步，以及程序当前所处的位置，因此在程序的调试阶段可以根据调试信息找到程序发生错误的地方以及错误的类型，从而修改程序。

11.4.9 shell 脚本程序格式

shell 程序是一种脚本语言，是一组命令、函数、变量或是任何在 shell 下可单独使用的内容。shell 脚本程序以纯文本形式存储在文件系统中，用户根据需要可以执行它，但它的执行是在 shell 的控制和解释下进行的。shell 程序在 DOS/Windows 系统中被称为批处理程序，这种称呼是比较结合实际的，因为编写 shell 程序的目的就是要成批地处理一些任务，比如某系统的安装、配置和启动等，故在 UNIX/Linux 系统中也有 batch 命令，shell 编程的目的也在于此。

一个 shell 程序内可以包含长串命令、复杂控制结构、系统变量、用户变量和用户函数、信号处理等。如何组织或者说按什么格式书写 shell 脚本程序，总体原则是，在 shell 提示符下的各种合法输入都可作为 shell 脚本程序的内容，再加上函数、结构控制语句和命令分组等就可以了，但在细节上还应注意以下几点：

1．注释行

使用“#”可引导注释，但若“#”不是出现在行首，则其前至少要有一个白空格。

2．第一行

bsh 或 bash 脚本程序的第一行建议是：

```
#!/bin/sh               或
#!/bin/bash
```

用于告诉系统这是一个 bshell 脚本程序，要它用 sh 或 bash 来解释执行。对于默认为 bshell 或 bashell 的用户来说，如此的第一行内容可以省略，但建议不要省略。实际上这也是一种注

释，因为它在脚本的正常执行过程中并不起作用。

脚本文件的第一行内容常用作对文件类型的描述，它将影响 file 命令对文件类型的判断，不仅如此，当为脚本赋予执行权，系统可以首行内容调用相关的程序解释执行之。因此，对于其他脚本也是这样要求的，比如 cshell、perl、python、sed 和 awk 脚本程序的第一行建议分别为：

```
#!/usr/bin/csh
#!/usr/bin/perl
#!/usr/bin/python
#!/usr/bin/sed
#!/usr/bin/awk
```

必要时，还可以在行末添加所需的参数（参见 sed 和 awk 示例），路径可能因系统而异。

3．继续行

shell 脚本的一行可以写得很长，但为了便于阅读和调试,建议不要写得太长，要在一个合适地方换行而使用继续行。为了将两个或几个逻辑上为一行的内容连接起来，除最后一行外的每个不完整行末必须以“\”结束，且“\”必须为最后一个字符，也就是说其后不能再有除回车以外的任何字符，空格也不行。

4．分号

在 shell 脚本中，可以将几个短行写在同一行内，但中间应使用分号“;”分隔，比如

```
x=1
y=2
```

可以写为：

```
x=1; y=2
```

5．白空格与行缩进

在 shell 脚本中，除了字符串常量或变量值中的白空格是原样处理外，其他位置的空格仅仅是分隔符，不论多少个只算一个，起分隔作用。因此，可以在行首或任何起分隔作用的地方插入一些白空格使脚本内容更易于阅读。

6．白空格与运算符

在 shell 脚本中，所有运算符与它所操作的对象间都要使用白空格分隔，否则将造成“拼写”错误。例如，求 x 与 y 和的加运算必须写成“x_+_y”而不是“x+y”。再如，用于判断的方括号必须与其前后及其中的内容间保留空格，用于判断 x 是否等于 y 的式子可写成：

```
if _[_$x_=_$y_]; then …            #此处的“_”表示空格，为可视而设
```

而不能写成：

```
if[$x_=_$y]; then …               #错误：'['前后和']'前无空格   或
if _[$x=$y];then …                #错误：'['后、']'前和'='前后均无空格
```

还有，用于赋值的等号前后都不能有白空格。像这样与白空格相关的情况很多，不能一一列举，请读者在书写脚本程序时注意。

7．I/O 重定向与/dev/null

shell 是批处理过程，通常要执行多个逻辑上相关的命令。在 shell 脚本的执行过程中，进程间的联系往往是通过信号、管道或上一命令的返回值等来进行的，各独立进程的标准输出或标准错误对本 shell 程序可能是“无用”的并且会扰乱屏幕。因此，对无用的信息可通过 I/O 重定向的方法将它们重定向到空设备/dev/null 而废弃掉，这样做的最大好处是不需要启动 I/O。

8．临时文件的使用

对于通过标准 I/O 相关联的先后进程间的联系可以通过管道或命令替换来处理，只有在非常必要时才使用临时文件。对于临时文件的命名要尽量和当前进程相关，比如可将当前进程号$$作为文件名的一部分，这样可以避免同一 shell 脚本同时运行时造成的数据冲突。

可以使用 mktemp 命令创建系统内唯一的临时文件名，文件名模板形式为 temp.XXXXXX，前缀 temp 由用户自行定义，后缀 XXXXXX 不可改变。若以当前进程的 PID 作为前缀，文件名模板可写为$$.XXXXXX；若以当前进程的进程名作为前缀，文件名模板可写为$0.XXXXXX。比如

```
tmp_file=`mktemp   $$.XXXXXX`
```

将创建一个长度为 0 的临时文件，文件名存放在变量 tmp_file 中，至于文件名的具体内容取决于具体的操作系统。

在程序结束前，要主动删除自己创建的临时文件。程序可以设计为在正常结束时或逻辑执行的最后部分、退出前删除自己的临时文件。但并不能排除，程序可能会因收到信号而非正常终止，导致程序结束后它的临时文件却仍留在文件系统中。为了减少此类事情的发生，可以采用捕获信号的办法来处理：当某些信号到来后删除自己定义的临时文件。针对以上临时文件可按以下方法捕获信号并删除临时文件：

```
trap 'rm -f $tmp_file; ' 1 2 3 15
```

这样可以保证在收到信号 1、2、3 或 15 时删除临时文件$tmp_file。

9．非法字符及处理

由于操作系统间文本文件的格式不同，当文本在不同操作系统间复制时，可能会导致格式错误。比如 DOS/Windows 系统下文本文件的换行由“回车+换行”，即\r\n 或十六进制的 0x0D+0x0A，但在 UNIX/Linux 下换行只由一个字符 0x0A 构成，这样当 DOS/Windows 下的文本文件复制 UNIX/Linux 系统时，在每行的行末就多了一个字符 0x0D（即\r 或^M），而这个字符是 shell 解释程序所不能识别的。设使用 Windows 记事本编辑的文本文件 dossh，其内容为：

```
x=1
y=3
echo `expr $x + $y`
```

当在 UNIX/Linux 系统执行时，会出现以下错误：

```
expr: non-numeric argument
```

而这个错误就是由行末的不可见字符造成的。通过带-b 选项的 vi 命令编辑此文件时就可看到该字符的庐山真面目了。输入命令：

```
vi -b dossh
```

进入编辑界面后看到：

```
x=1^M
y=3^M
echo `expr $x + $y`^M
```

为了使程序得到正确执行，必须删除其中多余的字符。删除字符^M 的办法可采用多种方式，以下是其中的两种：

（1）使用带有-b 选项的 vi 命令编辑文件，在 vi 的命令状态下执行命令：

```
:g/\r/s///g
```

删除^M 后存盘。

（2）在 shell 命令提示符下执行命令：

```
#tr -d ’\015’ < dossh > dossh1          或
#tr -d '\r' < dossh > dossh1
```

将删除^M 字符后的内容保存到 dossh1。

实际上，shell 对 shell 脚本程序并没有非常严格的格式要求，只要在内容组织上符合 shell 命令格式和逻辑控制的要求，在每行长度上不超过单行所允许的最大长度就可以了。

习题 11

1．思考题

（1）正则表达式可用于模式匹配与搜索，常见的正则表达式有几类？BRE 的正则表达式可完全使用于 ERE 吗？

（2）shell 是解释语言还是编译语言？

（3）awk 的默认域分隔是什么？如何改变 awk 的域分隔符？

（4）在 shell 中如何回到刚离开的工作目录？

2．填空题

（1）grep -E '[Hh]enr(y|ietta)' file 的功能是（____________）

（2）grep –v "^#" /etc/syslog.conf 的作用是（____________）

（3）sed -e 's/sysman/System Manager/g' <ifile >ofile 的作用是（____________）

（4）sed '/^$/d;/[:space:]*$/d;' <ifile >ofile 的作用是（____________）

（5）awk -F: '{ print $1 } END { print NF; print NR }' /etc/passwd 的作用是（__________）

（6）awk　'{ print toupper($0) }' <ifile　> ofile 的作用是（____________）

3．综合题

（1）试分别用 grep、sed 和 awk 实现：删除文件 ifile 中以#开始的注释行，并将结果输出文件 ofile。

（2）设计一个菜单 shell 程序，要求：

① 捕获信号 1、2、3、15。

② 当收到信号后在终端上显示：” I Received Signal #”，其中#为收到的信号编号。

③ 其他仿 sh03.sh。

（3）设有 shell 程序，内容为：

```
#!/bin/bash
pkg="tcsh"
hostnamectl | grep -i Ubuntu >/dev/null 2>&1
if [ $? -eq 0 ]; then
        cmd_q='dpkg -l '; cmd_i='apt    install '
else
        cmd_q='rpm -q '; cmd_i='yum    install '
fi
$cmd_q $pkg    >/dev/null 2>&1
if [ $? -eq 0 ]; then echo "Package: $pkg has been installed!"
else      $cmd_i ${pkg}
fi
```

试写出程序的功能和执行的可能结果。

（4）在 UNIX/Linux 系统中，环境变量非常重要，在字符界面下试着说明：

① PATH 变量的作用，如何得到 PATH 的值？

② 在 UNIX/Linux 系统超级用户的环境变量 PATH 中允许包含当前目录吗？

③ 若在超级用户的当前目录内有可执行文件 mypro，如何执行它（写出执行时键盘输入的内容）？

④ 如何使用该命令在后台执行（写出执行时键盘输入的内容）？

（5）设计一个 shell 程序计算 n 的阶乘。要求：

① 从命令行接收参数 n；

② 在程序开始后立即判断 n 的合法性，即是否有参数，若有是否为正整数，若非法请给出错误提示；

③ 最后输出计算的结果。

实验 11

1．sed 和 awk 的使用。

2．shell 编程与调试。

3．修改 sh03.sh，使之能作为某用户的“登录 shell”，也就是说，在修改用户.bash_profile（红帽）或.profile（Ubuntu）后，使得用户：

（1）在字符界面登录成功时直接进入操作界面；当用户输入 0/q/Q 退出时，直接回到登录界面。

（2）在图形界面登录时不起作用（也就是说不影响图形界面的使用）。

4．为 chroot 编写通用的准备 bshell 脚本程序，即对于 chroot 的用法为：

```
chroot   newroot   [cmd]
```

为命令 cmd 或$SHELL 提供 chroot 方式运行的准备脚本。

第 12 章　C/C++及其他编程环境

12.1　编译器

12.1.1　功能及用法

Linux 系统的 C 编译器为 gcc，它是 GNU 推出的功能强大的编译工具，因为 UNIX 系统的编译器为 cc，所以在 Linux 系统中还保留一个链接 cc，用于和 UNIX 的向后兼容。Linux 的面向对象的 C 编译器为 g++或 c++，它不仅功能强大、结构灵活，且可以通过不同的可选模块来支持多种语言，如 Java、Fortran77、Ada 和汇编等。gcc 能将 C/C++源程序和目标程序编译并调用链接程序 ld 生成可执行文件，如果用户没有给出可执行文件的名字，gcc 将默认生成一个名为 a.out 的可执行文件。

在 Linux 下可以使用 vi、vim 或 emacs 等任意一种文本编辑工具编写 C 的源程序。

在 UNIX/Linux 系统中，可执行文件没有统一的后缀或扩展名，系统通过文件的权限和内容来确定一个文件是否可执行。gcc/g++需要通过后缀来区别输入文件的类别，其所遵循的部分扩展名约定如表 12-1 所示。

表 12-1　gcc/g++文件扩展名约定

扩　展　名	文 件 类 型	扩　展　名	文 件 类 型
.c	C 程序源代码	.S, .sx	未预处理的汇编程序
.C,.cp,.cc,.c++,.cpp	C++程序源代码	.h	头文件
.i	预处理后的 C 源代码文件	.o	目标文件
.ii	预处理后的 C++源代码文件	.a	静态库文件
.m	OC 程序源代码	.so	共享库文件
.M, .mm	OC++程序源代码	.s	预处理后的汇编程序
.mi	预处理后的 OC 源代码文件	.mii	预处理后的 OC++源代码文件

gcc 的一般用法为：

```
gcc/g++ [options] <filenames>
```

其常用格式为：

```
gcc/g++ [-c] [-S] [-E] [-s] [-g] [-static] [-shared] [-rdynamic]
        [-Idir …] [-Ldir …] [-lname] [-fpic] [-fPIC] [-Olevel]
        [-Dmacro[=defn] …] [-Umacro] [-mmachine-option …]
        [-o out_file] infile …
```

在我们介绍内容范围内，g++和 c++与 gcc 和 cc 使用相同的参数或选项，或者说 g++、c++、gcc 和 cc 有相同的用法。

12.1.2　参数及说明

gcc/g++的部分参数如表 12-2 所示。

表 12-2　gcc 的部分参数

参　　数	意　　义
-c	编译后仅输出*.o 类型的目标文件，而不链接生成可执行程序
-S	编译后仅生成汇编语言文件*.s，但不生成目标文件和可执行代码
-s	生成可执行文件时，删除符号表和重定位信息，生成成品软件时使用
-E	在预处理过程后结束，不进行编译和链接，也不生成可执行代码
-g	在可执行文件中加入调试信息，便于程序的调试
-Idir	将目录 dir 添加到头文件搜索范围
-lname	链接时搜索库 libname.a
-Ldir	将目录 dir 添加到库文件搜索范围
-o outfile	指定输出文件名，若不指定则采用默认方式
-mcpu=cpu-type	生成与机器相关的汇编代码
-O[L]	编译时进行优化。L 为优化级别，分别为 0～3 和 s，生成最终产品时使用
-static	禁止使用共享库（动态链接库）
-shared	生成共享库
-rdynamic	链接时使用共享库
-Dname[=val]	宏定义变量 name[=val]
-Uname	取消宏定义变量 name
-fpic/-fPIC	生成位置独立的代码（PIC，Position Independent Code），构造共享库时使用

12.1.3　示例

1．C 程序

设有一个 Hello World 程序，其文件名为 hello.c，其内容如下所示。

```
#include    <stdio.h>
int main()
{
        printf("Hello C World!\n");
}
```

可按以下方法进行编译，以生成相应的结果：

```
$ gcc hello.c                  #生成可执行程序 a.out
$ gcc -o hello hello.c         #生成可执行程序 hello
$ gcc -c hello.c               #生成目标文件 hello.o
$ gcc -S hello.c               #生成汇编语言程序 hello.s
$ gcc -S -mcpu=i386            #生成 80x86 格式汇编语言
```

编译生成可执行文件之后，就可以运行了，方法为：

```
$ ./hello        或
$ ./a.out
```

输出为：

```
Hello C World!
```

2．C++程序

gcc 可同时用来编译 C 和 C++程序，但是，gcc 命令可编译 C++源程序，而不能自动使用 C++的库进行连接。可使用 g++命令来完成 C++程序的编译和连接，它会自动完成目标代码与 C++库的链接并生成可执行程序。下面是一个 C++版的 Hello World 程序，其文件名为 hello.c++，内容为：

```
#include    <iostream>
using namespace std;
main()
{
        cout<<"Hello, C++ World!" << endl;
}
```

我们可以像使用 gcc 一样，使用 c++或 g++来编译 hello.c++：

```
$ g++ hello.c++                #生成可执行程序 a.out
$ c++ -o hello hell.c++        #生成可执行程序 hello
$ g++ -s -o Hello hello.c++    #生成删除符号表的可执行程序 Hello
```

也可以使用 gcc 并指定库文件来编译 C++程序：

```
$ gcc -c hello.c++             #生成目标文件 hello.o
$ gcc -o h hello.c++ -lstdc++  #指定标准 C++库，生成可执行程序 h
```

3．多模块工程

设有一个开发项目，由 3 个 C 语言模块文件 f1.c，f2.c 和 f3.c 组成，它们的内容分别如下所示。

f1.c 源代码：

```
#include    <stdio.h>
int f1(int arg)
{
        printf("F1: you passed: %d\n",arg);
}
```

f2.c 源代码：

```
#include    <stdio.h>
int f2(char *arg)
{
        printf("F2: you passed: %s\n", arg);
}
```

f3.c 源代码：

```
#include    <stdio.h>
#include    <stdlib.h>
int f1(int );
int f2(char *);
int main()
{
        int tmpi=16;
        fprintf(stderr,"Begine:\n");
        f1(tmpi);
        f2("Hello World!");
        fprintf(stderr,":End\n");
        exit(0);
}
```

可以采用各模块文件分别编译然后再统一链接的办法进行编译。

```
$ gcc -c f1.c f2.c                    #生成 f1.o 和 f2.o
$ gcc -o f f3.c f1.o f2.o             #生成 f
```

也可以将所有模块源代码一起编译

```
$ gcc -o fp f3.c f1.c f2.c            #生成 fp
```

12.1.4 gcc/g++的工作过程

使用 gcc/g++由 C 源代码文件生成可执行文件的过程有以下四个阶段：预处理（也称预编译，Preprocessing）、编译（Compilation）、汇编（Assembly）和链接（Linking）。

在预处理阶段，调用 cpp 对源代码文件中的头文件、宏定义等进行处理，并进行语法、词法等分析，若无错误则生成中间文件。接着调用 gcc 进行编译，这个阶段根据中间文件生成以.s 为后缀的汇编文件。第三阶段调用汇编语言 as 对中间文件进行汇编，生成以.o 为后缀的目标文件。最后，当所有目标文件都生成之后，调用 ld 来完成链接工作，生成可执行程序。

用户可以根据自己的需要让 gcc/g++在编译的任何阶段停下来，以便检查或使用编译器在相应阶段的输出，或者对最后生成的二进制文件进行控制。

12.2 头文件

在 UNIX/Linux 的 C 中规定，默认头文件位置为/usr/include 及其子目录。依赖 UNIX 和 Linux 操作系统的头文件一般在/usr/include/sys 和/usr/include/linux 内。若不按默认规定存放头文件，则使用时必须用-Idir 指定位置，否则编译器无法找到它们。

在标准 C 中有两种形式的头文件使用方式：

（1）#include <headfile.h>

（2）#include “headfile.h”

两者的区别是#include <headfile.h>类型头文件搜索范围为默认位置/usr/include，而

#include“headfile.h”类型头文件的搜索位置为当前目录，在 Linux 的 GNU C 中，若当前目标不存在 headerfile.h，则也会到默认位置去搜索。

12.3 链接器与库文件

UNIX/Linux 的链接器为 ld，它的功能是将目标文件或库文件链接在一起，生成可执行文件，一般在编译过程的最后执行。

ld 有很多参数，这些参数可以直接通过 ld 命令在其命令行上使用，但在直接使用 ld 链接目标程序生成可执行程序时可能会遇到问题，更多的是通过编译器 gcc 或 g++的参数来控制 ld，部分参数说明参见表 12-2。

标准库文件一般存放在目录/lib 或/usr/lib（64 位系统下可能是/lib64 或/usr/lib64）。默认情况下链接器查找 C 语言的标准库函数。如果使用的不是标准的 C 库函数，必须通过-lib 或-libdir 指定库名和库文件的存放位置，否则将无法找到它。

库文件命名必须遵守一定命名规则，库文件名字必须永远以 lib 开头，后紧跟库名，文件名的后缀为

.a：传统静态库
.so：共享库或动态链接库

例如，libc.a 为标准 C 库，libstdc++.a 为 C++标准库，libm.a 为数学运算静态库，libm.so.6、libc.so.6 和 libstdc++.6 分别为数学运算、标准 C 和 C++库。需要说明的是，在操作系统级，静态库使用的越来越少了。

12.4 静态库

静态库也叫档案（archive），以.a 为后缀，用于编译链接后生成静态可执行文件。用户可以使用库管理程序 ar 和 ranlib 来创建和管理自己的或已有的静态库。

12.4.1 静态库的管理

静态库的管理命令有 ar、ranlib 和 nm。

1）静态库管理命令（ar）

ar 用于静态库文件的管理，其功能是库创建、修改和从库中取出模块等，其部分参数如表 12-3 所示。

表 12-3　ar 的部分参数

类　　型	参　　数	意　　义
功能参数	d	从库文件中删除模块
	m	移动模块位置，与 a、b 和 i 配合使用
	p	显示模块的内容到标准输出
	q	快速追加模块到库文件的尾部，不检查是否有重复模块
	r	在库文件中插入模块，插入过程中替换重名模块

（续表）

类　　型	参　　数	意　　义
功能参数	t	列出库文件中模块的列表
	x	从库文件中取出模块
修饰参数	a	在指定模块后追加
	b,i	在指定模块前插入
	c	创建库文件
	s	为库文件建立或更新索引。ar -s libfile 与 ranlib libfile 相同
	v	显示工作过程信息

2）为静态库建立索引（ranlib）

ranlib 用于为刚建立的库文件建立索引表，通过索引表可以加快库文件搜索速度。ranlib 的用法为：

```
ranlib [-hvV] ar_file
```

ranlib libfile 等同于 ar -s libfile。

3）列出库文件中的符号表（nm）

命令 nm 用于列出库文件或目标文件中的符号表，其用法为：

```
nm [options] [objfile ...]
```

若不提供 objfile，默认为 a.out。nm 默认按符号类型进行排序输出，选项-r 可让其逆序输出，-p 让其不排序输出。

12.4.2　构造静态库

这里，我们以多模块工程为例，使用 ar 构造自己的静态库。由于库文件中是不能包含 main()函数的，故只使用 f1.c f2.c 来构造。设静态库名为 libmyl.a，构造过程如下。

```
$ cc -c f1.c f2.c                    #生成目标文件 f1.o 和 f2.o
$ ar crv libmyl.a f1.o f2.o          #生成库 libmyl.a
$ ranlib libmyl.a                    #为子函数建立索引表
$ nm libmyl                          #查看索引表
```

12.4.3　使用自己的库

当用户创建自己的静态库之后，就可以按照使用系统库的方法来使用它。以下为使用静态库的实例：

```
$ cc -o fp f3.c libmyl.a          #使用库 libmyl.a 和 f3.c 生成可执行程序 fp
$ cc -o fp f3.o libmyl.a          #使用库 libmyl.a 和 f3.o 生成可执行程序 fp
$ cc -o fp f3.c -L. -lmyl         #-L.指定当前目录，-lmyl 指定静态库文件 libmyl.a
```

若将 libmyl.a 放在系统的标准位置，比如/lib 或/usr/lib，则可像其他库一样使用。假设已

经存放到了标准位置，使用方法为：

```
$ cc -o fp f3.c -lmyl          #使用库 libmyl.a 和 f3.c 生成可执行程序 fp
$ cc -o fp f3.o -lmyl          #使用库 libmyl.a 和 f3.o 生成可执行程序 fp
```

12.5 共享库

Linux 系统的另一种库文件为共享库，用于生成动态链接的可执行程序，其标准位置与静态库相同。共享库文件名的格式为：

```
libNAME.so.N
```

NAME 为库名，N 为版本号。可用 ldd 和 ldconfig 命令管理共享库。

共享库的搜索路径为环境变量 LD_LIBRARY_PATH 指定的位置或库文件的标准位置。

12.5.1 共享库构造

共享库构造非常简单，只需要在生成目标文件时使用-fpic 选项，在构造库的时候使用-shared 参数就可以了。

我们仍使用 12.1.3 节多模块工程中的 f1.c 和 f2.c 构造共享库，方法是：

```
    gcc -fpic -c f1.c f2.c                    #生成目标文件。注意：这里使用了选项-fpic
    gcc -shared -o libmy.so f1.o f2.o         #由目标文件生成共享库
或  gcc -shared -o libmy.so -fPIC f1.c f2.c   #由源文件生成共享库
```

12.5.2 共享库的管理

1．显示库间依赖关系（ldd）

ldd（Library Dependency Display）用来显示一个可执行程序或共享库所使用的共享库间的依赖关系。例如：

```
#ldd a.out
#ldd libmyl
#ldd `which bash`
```

2．ldconfig

ldconfig 的功能是在默认库目录（/lib 和/usr/lib）或动态库配置文件/etc/ld.so.conf 内所列的目录或指定目录下搜索共享库，进而创建动态装入程序 ld.so 或 ld-linux.so 所需的链接和缓存文件。缓存文件默认为/etc/ld.so.cache，其中保存有已排好序的动态链接库列表。

为了让系统或用户动态链接库能更好地被共享，需要运行 ldconfig 来对共享库进行配置。ldconfig 通常在系统启动时运行，而当用户安装了一个新的动态链接库时，需要手工运行这个命令，其用法为：

```
ldconfig [选项 …]
```

ldconfig 命令的部分参数如表 12-4 所示。

表 12-4 ldconfig 命令的部分参数

参　　数	意　　义
-n	仅扫描命令行指定的目录，不扫描默认目录（/lib，/usr/lib），也不扫描配置文件/etc/ld.so.conf 所列的目录
-f conf	指定配置文件为 conf，而非默认的/etc/ld.so.conf
-C cache	指定缓存文件为 cache，而非默认的/etc/ld.so.cache
-r root	改变并使用 root 作为根目录（通过调用 chroot 来实现）
-l	专家模式。在通常情况下，ldconfig 搜索动态链接库时将自动建立动态链接库的链接。选择此项时，由用户手工设置链接
-p	显示当前缓存文件内的所有共享库
-v	显示工作时正在扫描的目录及共享库，还有它所创建的链接名

ldconfig 的使用示例如下：

（1）显示搜索的目录和共享库，并更新缓存文件/etc/ld.so.cache。

```
ldconfig -v
```

在默认情况下，ldconfig 不输出任何东西。使用-v 参数可以显示正在扫描的目录及搜索到的共享库。

（2）安装共享后，更新目录/lib 内的符号链接。

```
ldconfig -n /lib
```

12.5.3 共享库编程

共享库的使用与静态库有所不同，要使用头文件 dlfcn.h 和几个相关的函数：dlopen、dlsym、dlclose 和 dlerror。

1．共享库打开（dlopen）

dlopen 用于打开指定共享库，并返回文件描述符，其原型为：

```
void *dlopen(const char *filename, int flag);
```

dlopen()失败时返回 NULL 值，否则返回文件描述符。

变量 filename 为共享库名。若文件名不带有路径，将按以下顺序搜索库文件：

（1）环境变量 LD_LIBRARY_PATH 值所指的路径。

（2）动态链接缓冲文件/etc/ld.so.cache。

（3）库文件默认目录/lib 和/usr/lib。

变量 flag 用来表示在什么时候解决未定义的符号，其取值范围与意义如下：

（1）RTLD_LAZY 指定在动态链接库的函数执行时解决。

（2）RTLD_NOW 指定在 dlopen 返回前就解决所有未定义的符号问题。一旦有未解决好未定义的符号，dlopen 将返回 NULL 表示错误。RTLD_LAZY 和 RTLD_NOW 可以与 RTLD_GLOBAL 配合使用，使得那些在以后才加载的库可以获得其中的符号。

2．在共享库中查找函数等符号（dlsym）

dlsym 用于返回共享库中指定函数或符号的入口地址，其原型为：

```
void *dlsym(void *handle, char *symbol);
```

dlsym 根据共享库文件描述符（handle）与符号（symbol），返回 symbol 对应的（函数）入口地址，相当于返回一个（函数）指针。

3．共享库关闭（dlclose）

dlclose用于关闭已经打开的指定共享库文件，此操作应在共享库相关操作完成之后进行，其原型为：

```
int dlclose (void *handle);
```

4．共享库错误处理（dlerror）

函数 dlerror()用于返回共享库操作状态信息。当共享库操作函数执行失败时，dlerror 可以返回出错信息，否则返回值 NULL 表示成功，其原型为：

```
const char *dlerror(void);
```

12.5.4　共享库使用示例

为了使用刚创建的共享库，需要对 12.1.3 中多模块工程的文件 f3.c 进行修改。假定修改后的文件被命名为 f3n.c，其代码如下所示。

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#define SO_FILE "libmy.so"
main( ){
        void *sfp;  char *err;
        int     tmpi=16;
        int (*f1)(int ),(*f2)(char *);              //定义函数指针

        sfp=dlopen(SO_FILE,RTLD_LAZY);              //打开共享库
        if(sfp==NULL){
            fprintf(stderr,dlerror()); exit(1);
        }
        f1=dlsym(sfp,"f1");                         //获取函数 f1 入口地址（指针）
        err=dlerror();                              //检查是否成功
        if(err){
            fprintf(stderr,err); exit(2);
        }
        f2=dlsym(sfp,"f2");                         //获取函数 f2 入口地址（指针）
        err=dlerror();                              //检查是否成功
```

```
            if(err){
                fprintf(stderr,err); exit(3);
            }
            fprintf(stderr,"-----------begine-------------\n");
            f2("Test String");                          //调用函数 f2
            f1(tmpi);                                   //调用函数 f1
            fprintf(stderr,"++++++++++++end+++++++++++++++\n");
            dlclose(sfp);                               //关闭共享库
            exit(0);
    }
```

编译方法为：

```
cc -o mydyl f3n.c -ldl
```

选项-ldl 则告诉链接程序 ld 使用 dl 函数库，由共享库 libmy.so 生成可执行程序 mydyl。

编译也可以使用-rdynamic 参数，告诉链接程序在链接时所有函数均使用共享库，其方法为：

```
cc -rdynamic -o myp f3n.c -ldl
```

由于在 f3n.c 中共享库文件名没有带路径，故共享库 libmyl.so 必须放在标准位置或通过以下方法设置环境变量 LD_LIBRARY_PATH 的值：

```
$ export LD_LIBRARY_PATH=MYPATH:$LD_LIBRARY_PATH
```

MYPATH 为共享库 libmyl.so 所在的位置，若为当前目录，可写为：

```
$ export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH                              #或
$ export LD_LIBRARY_PATH=`pwd`:$LD_LIBRARY_PATH
```

12.6 make 与 Makefile

在现代的编程环境中，各种集成开发环境已为编程者考虑得很周到，当一个软件或工程或项目中的一个文件或一部分被改变后，再重新编译或链接时，用户不需要特殊控制，开发系统就会根据当前状况自动完成。UNIX/Linux 提供了 make 命令和 Makefile 文件来实现对开发项目的管理。

12.6.1 make 的用法简介

make 命令会根据 Makefile 的内容对项目进行管理。make 能自动确定哪一个模块被修改了，然后再进行统一、无遗漏、无重复的编译。

UNIX/Linux 系统中 make 使用的默认文件是 makefile，若 makefile 不存在，则查找 Makefile，也可通过-f file 参数指定文件名。make 的用法为：

```
make [-f filename] [options] [targets]
```

make 有很多参数，部分参数如表 12-5 所示。

表 12-5　make 命令的部分参数

参　数	意　义
-C dir	在读 Makefile 或做任何操作前切换到 dir 目录，一般用于对目录的递归搜索
-d	显示调试信息
-f file	指定 file 文件作为 Makefile，而非使用默认的 makefile 或 Makefile
-I dir	指定 Makefile 搜索目录
-k	在默认情况下 make 遇到错误将终止执行，-k 可以让其在出现错误时尽量工作长一些，以便观察分析
-n	让 make 在不真正编译的情况下列出将执行的步骤
-s	安静工作。默认情况下，每执行一条命令就要首先显示该命令。可通过在命令前加@的办法屏蔽本行输出

12.6.2　Makefile 文件

Makefile 文件的内容是描述项目或软件中模块之间的相互依赖关系以及目标文件、可执行程序产生时要执行的命令等。make 命令在工作时要依据 Makefile 文件中的规定来生成目标或可执行文件，并保证结果的正确性。

1．Makefile 文件包含的 5 类内容

Makefile 文件包含的内容有 5 类，见表 12-6。

表 12-6　Makefile 文件包含的 5 类内容

类　型	意　义
显式规则	显式告诉 make 如何编译或构造一个目标
隐式规则	隐式通过变量或规则告诉 make 如何编译或构造一个目标
变量定义	在 Makefile 中可以像 shell 编程一样定义和使用变量
命令	定义完成某任务所使用的命令，一般为 shell 命令
注释	#开始的部分，意义同 shell 编程

2．规则

规则中的项目定义必须从最左边开始，一个规则中的第二行以后的行必须以<tab>键开始。规则的格式如下：

```
targets ：prerequisites
<tab>commands
```

或

```
targets ：prerequisites；commands
<tab>commands
```

其中，targets 为目标；prerequisites 为目标所依赖的对象，简称对象；commands 为构造或生成目标所需执行的命令，若一个命令以@开始，则在 make 的执行过程中不会被显示，否则执行前将首先显示该命令，就像 shell 脚本程序执行时设置了-v 选项一样。

由上可见，规则是由目标（target）、依赖对象和达到目标所要执行的命令组成的。比如，f1.c 和 f1.o 的关系可以简单地描述为：

```
f1.o: f1.c
<tab>gcc -c f1.c
```

这里的 f1.o 为目标，它依赖于 f1.c，依赖方法为执行命令“gcc -c f1.c”，或者说，要实现此目标就要执行该命令。

这是直截了当地显示规则，有时为了完成整个工程或体现工程的局部，还可人工制造出一些具有特殊意义的规则，比如 all（整个工程）、install（安装）、uninstall（卸载）、clean（清理）等目标，作为工作的目标，这些通常被称为伪目标。

规则中的 commands 是达到目标所要执行的 shell 命令，其中也包括用于分支和条件判断元素，这些可以参照 shell 编程的方法来处理。

3．Makefile 文件中的常用符号

Makefile 文件中有很多用于控制的符号，部分符号见表 12-7。

表 12-7　Makefile 文件中的可使用的部分符号

符　号	意　义
$@	目标名
$*	隐匿规则中的词干（stem）；静态规则中的%；显式规则中的删除了后缀的目标名（若无后缀，则为空）
$%	当目标是库文件时，目标内的成员名。例如，目标 x.a（y.o）的目标名为 x.a，成员名为 y.o
$^	由空格分隔的所有依赖对象（去除重复者）
$+	由空格分隔的所有依赖对象（包含重复者）
$?	依赖对象中被更新者
$<	当前规则中的第一个依赖对象

表 12-7 中的变量也叫自动变量，它们随规则的改变而改变。

4．变量与通配符

Makefile 文件中可以像 shell 一样使用变量，且比 shell 中有更大的灵活性，比如定义变量时，等号的两边可有空格等。定义变量的等号有“=”或“:=”两种形式，主要作用是定义新变量或修改已经定义的变量或在已定义变量的基础上定义新变量，例如已知：

```
var1 = a.c b.c c.c
```

则

```
var2 = $(var1:.c=.o)
```

定义

```
var2 =a.o b.o c.o
```

而

```
var1 += d.c
```

重定义 var1，其值为

```
var1 = a.c b.c c.c d.c
```

需要说明的是，“:=”与“=”是有区别的，当使用“=”时，变量将做递归或扩展，而使用“:=”只进行简单替换。

Makefile 文件中还可以使用通配符“%”，目的是定义或重定义模式规则。例如：

```
%.o：%.c
<tab>gcc –c $<
```

定义一个规则：所有目标文件*.o 依赖 C 语言源程序*.c，且生成方法为 gcc -c $<。

12.6.3 Makefile 示例

1．多模块项目编译 Makefile 示例

在 12.1.3 中的多模块工程中有三个模块，分别是 f1.c，f2.c 和 f3.c，它们之间的关系如图 12-1 所示。按照 gcc 的工作过程，对模块的编译和链接过程如下所述。

（1）编译各源文件生成目标代码：

```
gcc -c f1.c
gcc -c f2.c
gcc -c f3.c
```

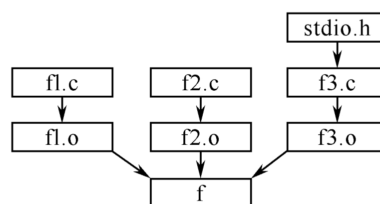


图 12-1　模块间关系示意图

（2）生成可执行程序：

```
gcc -o f f1.o f2.o f3.o
```

此过程按照 Makefile 文件的规定可表示为：

```
f1.o: f1.c                    #f1.o 依赖于 f1.c，生成过程为：
        gcc -c f1.c
f2.o: f2.c                    #f2.o 依赖于 f2.c，生成过程为：
        gcc -c f2.c
f3.o: f3.c                    #f3.o 依赖于 f3.c，生成过程为：
        gcc -c f3.c
f: f1.o f2.o f3.o             #f 依赖于目标文件 f1.o，f2.o 和 f3.o，其生成过程为：
        gcc -o f f1.o f2.o f3.o
```

按照 Makefile 的格式要求，可以构造此项目的 Makefile 内容，如下所示。

```
f: f1.o f2.o f3.o
        gcc -o f f1.o f2.o f3.o
f1.o: f1.c
        gcc -c f1.c
f2.o: f2.c
        gcc -c f2.c
f3.o: f3.c
```

```
        gcc -c f3.c
```

有了 Makefile 文件，可以使用 make 命令对此项目进行编译，编译方法为：

```
$ make                    #或
$ make f
```

若要编译单个项目，例如 f1.o，可以使用以下方法：

```
$ make f1.o
```

还可为此 Makefile 文件增加 all、clean、install 和 uninstall 目标，若此时 Makefile 文件记为 makefile_1，其内容如下所示。

```
all: f                          #make all to build f
f: f1.o f2.o f3.o
        gcc -o f f1.o f2.o f3.o
f1.o: f1.c
        gcc -c f1.c
f2.o: f2.c
        gcc -c f2.c
f3.o: f3.c
        gcc -c f3.c
clean:                          #clean all my object file f?.o
        rm -f f?.o f &>/dev/null
install: all                    #make all and install f to /usr/bin with permission 755
        install -m 755 /usr/bin
uninstall:
        rm -f /usr/bin/f
```

可以使用通配符对 makefile_1 进行简化，若简化后的文件记为 makefile_2，其内容如下所示。

```
all: f                          #make all to build f
f: f1.o f2.o f3.o
        gcc -o f f1.o f2.o f3.o
%.o: %.c
        gcc -c $<
clean:                          #clean all my object file f?.o
        rm -f f?.o f &>/dev/null
install: all                    #make all and install f to /usr/bin with permission 755
        install -m 755 /usr/local/bin
uninstall:
        rm -f /usr/bin/f
```

此时，编译整个工程的做法是：

```
$ make -f makefile_2                #或
$ make f   -f makefile_2
```

若要编译单个项目，例如 f1.o，可以使用以下方法：

```
$ make f1.o    -f makefile_2
```

编译并安装的命令是

```
$ make install -f makefile_2
```

卸载的命令是

```
$ make uninstall -f makefile_2
```

2．多模块项目共享库构造与使用 Makefile 示例

1）共享库创建

根据 12.5.1 共享库构造过程和 Makefile 文件的要求，可为共享库的构造过程编制 Makefile。设文件名为 makefile.somk，其内容如下所示。

```
%.o : %.c                              #定义目标文件生成规则
    gcc -fpic -c $<                    #目标文件生成命令（注意：使用了-fpic）
libmyl.so : f1.o f2.o                  #定义共享库生成规则
    gcc -shared -o $@ f1.o f2.o -ldl   #共享库生成命令（注意：使用了-shared）
```

据此，构造共享库可以使用以下命令来完成：

```
make libmyl.so -fmakefile.somk
```

为了应用的方便，我们对 makefile.somk 进行改造，为其添加更加灵活的变量定义和完成特定功能伪目标。设改造后的 makefile 名为 makefile.somk1，其内容如下所示。

```
myl=libmyl.so                          #定义目标名
dst=/usr/lib/$(myl)                    #定义安装位置
SRC=f1.c f2.c                          #定义源文件
OBJ=$(SRC:.c=.o)                       #定义目标文件
all : $(myl)                           #定义 all 目标（只编译，不安装）
%.o : %.c                              #定义目标文件生成规则
    cc -fpic -c $<                     #目标文件生成命令（注意：使用了-fpic）
$(myl) : $(OBJ)                        #定义共享库生成规则
    cc -shared -o $@ $(OBJ) -ldl       #共享库生成命令（注意：使用了-shared）
install:
    cp $(myl) $(dst)                   #仅复制 libmyl.so 到标准位置
uninstall:
    rm -f $(dst)                       #仅删除/usr/lib/libmyl.so
make_install: all install              #构造并安装
clean:                                 #定义清理目标
    rm -f $(OBJ)                       #仅清理目标措施
clean_all: clean                       #删除目标文件和 libmyl.so
    rm $(myl)
clean_uninstall: clean_all uninstall   #删除目标文件、libmyl.so 和/usr/lib/libmyl.so
list:                          #列出所有内容：源文件、目标文件、libmyl.so 和/usr/lib/libmyl.so
    ls -l $(SRC) $(OBJ) $(myl) $(dst)
```

在 makefile.somk1 中有 all、$(myl)、install、uninstall、make_install、clean_all、clean、

clean_uninstall 和 list 等目标。all 与$(myl)是相同的，功能是构造 libmyl.so；install 和 uninstall 分别只用于对共享库的安装与删除；make_install 用于共享库的构造与安装；clean 仅仅用于对目标文件的删除；clean_all 用于对目标文件和共享库文件的删除；clean_uninstall 则是彻底的清理，删除除源代码以外的所有文件；list 用于对源代码文件、目标文件、共享库文件和已经安装共享库文件的显示。用户可根据需要，继续添加其他功能目标。

2）共享库使用

对于 12.5.3 中的共享库使用示例，也可以为其编写一个简单的 makefile 文件。设 makefile 为 makefile.souse，其内容如下所示。

```
SRC = f3n.c                              #定义源文件
OBJ = $(SRC:.c=.o)                       #定义目标文件
TGT=mydyl                                #定义成果程序
all : $(TGT)                             #定义 all 目标
%.o : %.c                                #定义目标文件生成规则
    gcc -c $<                            #目标文件生成命令
$(TGT): $(OBJ)                           #定义成果文件生成规则
    gcc -o $@ $(OBJ) -ldl                #成果文件生成命令（注意：使用了-ldl 选项）
clean:                                   #定义清理目标
    rm -f $(OBJ)
clean_all: clean                         #删除目标文件和成果文件
    rm $(TGT)
```

生成目标文件可使用命令：

```
$ make -f makefile.souse f3n.o
```

必要时可先运行：

```
$ make clean -f makefile.souse           #或
$ make clean_all -f makefile.souse
```

使用共享库编译生成可执行程序 mydyl，可使用命令：

```
$ make -f makefile.souse                 #或
$ make all -f makefile.souse
```

需要时，还可为其增加 install、uninstall 等功能，用于结果程序的安装和卸载。

12.7 调试器（gdb）

程序调试是程序开发中的一个重要环节，通过调试可以找到程序中的问题。在 UNIX/Linux 系统里，调试工具为 gdb。

12.7.1 gdb 功能

gdb 是一个用来调试 C 和 C++程序的功能强大的调试器，它使用户能在程序运行时观察程序的内部结构、变量的值和内存的使用情况等。gdb 提供的功能有：监视或修改程序中变

量的值；设置断点以使程序在指定的代码行上暂停执行和单步执行或程序跟踪等。

12.7.2　gdb 基本命令

gdb 支持很多的命令，使用户能实现不同的功能。表 12-8 列出了 gdb 调试时部分常用命令。

表 12-8　gdb 部分常用命令

命　　令	功　　能	命　　令	功　　能
break	在代码里设置断点	step	执行一行源代码而且进入函数内部
bt	反向跟踪，显示程序堆栈	run	执行当前被调试的程序
continue	继续 break 后的执行	quit	退出 gdb
File	装入想要调试的可执行文件	set	设置变量的值
kill	终止正在调试的程序	shell	在 gdb 内执行 shell 命令
list	列出产生执行文件的源代码的一部分	print	显示变量或表达式的值
next	执行一行源代码但不进入函数内部	where	显示程序当前的调用栈

gdb 的命令在不引起歧义的情况下是可以缩写的，比如 list 可缩写为 l、kill 可缩写为 k、step 缩写为 s 等。同样在不引起歧义的情况下，可以使用 tab 键命令进行自动补全或查找某一类字符开始的命令，比如在 w 后输入 tab 键，可得到以下输出：

```
watch     where     while-stepping     whatis     while     ws
```

表示这些是 w 开头的命令。

12.7.3　程序调试方法

下面以 12.1.3 中多模块工程为例，介绍 Linux 系统内程序调试的基本方法。

1．编译时使用调试参数-g

编译时需要使用-g 选项，在生成的可执行文件中加入调试信息，方法是：

```
gcc -g -o myp f1.c f2.c f3.c
```

2．启动 gdb

```
gdb myp
```

3．查看源文件

```
(gdb) list                                    #首次使用 list，每次 10 行（可以设置）
1    #include    <stdio.h>
2    #include    <stdlib.h>
3    int f1(int );
4    int f2(char *);
```

```
5       int main()
6       {
7                   int tmpi=16;
8                   fprintf(stderr,"Begine:\n");
9                   f1(tmpi);
10                  f2("Hello World!");
(gdb) l                                        #再次使用 list，显示剩余部分
11                  fprintf(stderr,":End\n");
12                  exit(0);
13      }
```

4．设置断点

```
(gdb) break 9
Breakpoint 1 at 0x40066a: file f3.c, line 9.
```

5．开始执行

```
(gdb) run
Starting program: /root/myp
Begine:
Breakpoint 1, main () at f3.c:9
9                   f1(tmpi);
```

6．打印变量的值

```
(gdb)   print tmpi
$1 = 16
```

注：也可通过 set 命令设置变量 tmpi 的值，比如 set tmpi=100

7．单步执行，跟踪进入函数 f1（也可使用 next 命令执行 f1，而不进入函数 f1）

```
(gdb) step
f1 (arg=16) at f1.c:4
4                   printf("F1: you passed: %d\n",arg);
```

8．显示 f1()变量 arg 的值

```
(gdb) print arg
$2 = 16
```

9．可在任何位置使用 where 命令，显示程序的调用栈，得到自己的位置。方法是：

```
(gdb) where
#0   f1 (arg=16) at f1.c:4
#1   0x0000000000400674 in main () at f3.c:9
```

10．列出当前文件的内容

```
(gdb) list
```

```
#include    <stdio.h>
int f1(int arg)
{
        printf("F1: you passed: %d\n",arg);
}
```

11．程序继续执行直到结束

```
(gdb) continue
Continuing.
F1: you passed: 16
F2: you passed: Hello World!
:End
[Inferior 1 (process 2988) exited normally]
```

12．退出 gdb

```
(gdb) quit
```

12.8　UNIX/Linux 其他编程工具简介

以上介绍了 UNIX/Linux 系统的基本开发环境与方法，这是 C 语言开发与应用及其他开发的基础。尽管用户可以使用 C 基于相关的库开发应用程序或应用软件，但这样做对于开发某专业领域的软件是困难的，因为它需要专业环境、专业库和专业工具。本节只简单介绍 UNIX/Linux 系统的其他工具或开发环境，以供用户有其他编程需要时选择。

12.8.1　常用库与 GNOME/GTK 开发

1．XLIB

X Window（简称 X11）是建立在 C/S 网络模式基础上的图形子系统，是工作站普遍应用的窗口系统。X 系统带有一套低级的库函数，称为 xlib。xlib 提供了许多对 X 窗口的屏幕进行操作的函数，但是，直接使用 xlib 函数开发 GUI 应用程序是相当复杂的，因此还需要进一步的包装或提升。

2．GLIB

GLIB 是一种常用的 C 工具库，提供包括数据类型、类、线程支持、加载动态数据库和词库扫描等功能，还包括可移植性和一些实用程序功能，以及一些容器类，在设计 GDK 和 GTK 程序时要用到它。

3．GDK

GDK（Gimg Drawing Kit）是标准 xlib 函数库的一个基本封装，是用于实现图形用户接口的函数库。在一般情况下，如果要创建普通的图形接口应用程序，使用 GDK 就可以了。

4．GTK

GTK（Gimg Tool Kit）是GNOME应用程序中使用的GUI工具包，包括GNOME应用中的基础和可视化组件，包括字形、窗口、颜色和各种事件等，实质上是一个面向对象的应用程序接口。GTK+是GTK的增强版，没有平台限制，且越发展越强大，现在已经成为成熟的、易于使用的、轻便的、非桌面专用的工具包。它没有任何真正的桌面环境所要求的特点，可在不同操作系统间移植。现在系统中正在并行使用gtk+2和gtk+3版本。

5．GNOME

GNOME 的发布被分为多个小库，每个小库负责某个范围内的功能。主要的库是libgnome、libgnomeui和libgnorba等。libgnome执行GNOME需要的非GUI函数，如命令行分析，进程执行等；libgnomeui是GNOME围绕GTK+的包装和扩充，它提供许多支持，如对话框、制图技术、复合文档界面和大量有用的控件，也为停放菜单和工具栏增加了其他支持；libgnorba是一个围绕GNOME的CORBA（Common Object Request Broker Architecture，CORBA）功能的非常有用的包。

以上各库间的关系如图12-2所示。

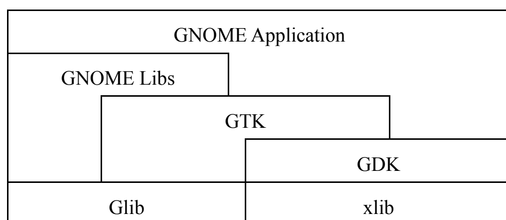


图12-2　GNOME应用程序与相关库的关系

基于以上库可以进行KDE、GDK和GTK/GNOME等开发，超出了本书的范围，不再介绍。

12.8.2　KDevelop/Qt开发

KDevelop是一个基于GPL的集成开发环境（IDE)，它最初是用于提供与KDE无缝式互操作的IDE及KDE基于Qt框架，因此提供了可以产生Qt图形界面的资源编辑程序。它提供编辑、编译、链接、排错、版本管理及计划管理等基本功能。

KDevelop的Qt提供图形界面的widget库，用于对象通信的signal-slot机制，使用事件循环和虚拟方法事件控制机制，其他辅助功能。

Linux自带KDevelop开发集成环境，安装后，用户可通过Application→Programming→KDevelop，或从CLI直接运行KDevelop命令打开操作界面，用户还可以通过设置来改变集成开发环境的界面外观。

12.8.3　eclipse

eclipse是一开源的、基于Java的可扩展平台，用于搭建开发平台，目前可以支持的语言

有 Java、C/C++、COBOL、PHP 和 Eiffel 等。用户可以使用 eclipse 的集成环境进行 Java 开发。使用前需要先安装之，方法如下：

```
#yum install eclipse-platform          #fedora
#apt install eclipse                   #ubuntu
```

对于红帽的 centOS 和 rhel，在发行版本没有提供可供直接安装的 eclipse rpm 包，用户需要从 http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/neon1a/下载安装包 eclipse-jee-neon-1a-linux-gtk-*.tar.gz。之后，将其解压到/opt，假设文件被下载到~/Download，方法是：

```
cd /opt
tar xvfz ~/Download/eclipse-jee-neon-1a-linux-gtk-*.tar.gz
```

这样，整个 eclipse 包就安装在了/opt/eclipse，eclipse 的主程序是/opt/eclipse/eclipse。用户可以在 CLI 界面下输入：

```
/opt/eclipse/eclipse
```

并回车后启动 eclipse。

为了应用方便，在/usr/bin 下为其创建一个符号链接，方法为：

```
ln -s /opt/eclipse/eclipse /usr/bin/eclipse
```

这样，再启动 eclipse 就不需要指定路径了。

还可以为其在桌面创建一个启动项，当需要时，从桌面通过鼠标来启动，方法是编辑/usr/share/applications/eclipse.desktop，使其内容为：

```
[Desktop Entry]
Encoding=UTF-8
Name=Eclipse
Comment=Eclipse Luna
Exec=/usr/bin/eclipse
Icon=/opt/eclipse/icon.xpm
Categories=Application;Development;Java;IDE
Version=1.0
Type=Application
Terminal=0
```

eclipse 启动方法是：

```
Application→Programming→eclipse          #或在 CLI 下执行命令
$ eclipse
```

12.8.4 Java 开发

Java 是由美国 SUN 计算机公司研究而成的一种语言，自从 1995 年发表以来，以其简单、跨平台和面向对象等优点，深受广大程序员的青睐，经过数年的迅速发展，已经成为和 C++等主流开发语言并列的语言。Java 以其安全性、结构独立性、可移植性、高效能、多线程和

动态性享誉编程界。近年来在移动开发、云计算、大数据、Web 开发与应用方面发挥了巨大作用。

在 Linux 系统中，有多种方法进行 Java 开发，常用的有 Eclipse 集成开发平台和 Java JDK 开发环境。

1. Java 运行和开发环境

Java 运行和开发需要 Java 运行和开发环境的支持。现在的 Linux 都提供 Java 开发和运行所需的软件包，需要时，可先安装它们，也可在安装系统时安装它们。在不同系统下的安装方法有所区别，在红帽系列系统和 ubuntu 系统的安装方法如下：

```
#yum install java-*-openjdk-devel -y          #红帽系统
#apt install default-jdk -y                   #ubuntu 系统
```

2. Java 示例

安装之后，用户就可使用它编译和运行 Java 程序了。设有一个 Java 版的 hello world 程序 hello.java，代码如下：

```
public class hello {
    public static void main( String []argc) {
        System.out.println("Hello JAVA World!");
    }
}
```

编译和运行 hello.java 方法如下：

```
$ javac hello.java      #编译
$ java hello            #运行
```

12.8.5 Perl 开发

Perl（Practical Extraction and Report Language）是由 Larry Wall 设计的，并由他不断更新和维护，但现在是由 Perl 社区维护的。顾名思义，Perl 是实用摘录和报表语言，这是 Perl 设计的初衷，但随着进一步的发展，它的功能大大增加，被广泛应用于各个领域，从航天工程到分子生物学，从数学到语言学，从图形处理到文档处理，从数据库操作到网络管理等。

Perl 最初是作为一种 UNIX 的黏合语言设计的，但很快就被移植到了大多数其他操作系统上，所以 Perl 几乎可以在任何“地方”运行，甚至同一 Perl 脚本可以不加修改地在不同操作系统上工作。

Perl 是开源的，是一种易学易用语言，但也是一种特性非常丰富的语言，且具有高级语言的强大功能和灵活性。Perl 被设计成可以通过模块进行扩展，用户可以使用 Perl 快速设计、编写、调试 Perl 应用程序，还可以在需要时扩展自己的应用程序。用户可以在其他语言中嵌入 Perl 程序，也可以在 Perl 程序中嵌入其他语言，还可以通过模块导入机制，把外部的扩展当成 Perl 的内置特性，而且那些面向对象的语言模块在 Perl 内仍保持其面向对象特性。

Perl 是一种脚本语言，它的解释器是 Perl。但 Perl 脚本的执行过程不完全与 shell 脚本相

同，首先将脚本程序“编译”成内部格式代码，还可以进行优化，并将所遇到的错误反馈给用户，一经编译通过，这些中间代码就会被立即执行。

Perl 脚本的执行需要 Perl 软件包的支持，用户可在系统安装时安装它们，也可在系统扩充时安装它们。

以下是一个简单的 Perl 脚本：

```
#!/usr/bin/perl
print "Hello Perl World!\n";
print "Name Please:";
$name=<stdin>;
chop($name);
print "your name is \"$name\", Your are welcom!\n";
```

perl 脚本程序的扩展名为.perl 或.pl。假设此脚本文件的文件名为 hello.pl，如果第一行

```
#!/usr/bin/perl
```

设置正确，可在先为其增加执行权后，再按 shell 命令方式运行它，方法是：

```
$ chmod   +x   hello.pl;
$ ./hello.pl
```

如果第一行设置不正确，就只能使用 Perl 命令直接解释并执行之，方法为：

```
$ perl hello.pl
```

12.8.6 数据库开发

在 Linux 系统中可以使用数据库的选择较多，其中 mSQL、MySQL、PostgreSQL 和 SQLite 是最流行的、开源的和基于 SQL 的数据库，并且 MySQL 和 PostgreSQL 的许可证也是免费的。还有一个正式商业产品 InterBase，而它的 Linux 版是免费的，也是完全开放源代码的。除此之外，用户还可使用商业数据库，比如 Oracle、Sybase、Informix 和 DB2 等。

mSQL（miniSQL）数据库是由澳大利亚大学从事网络工程的博士研究生 David Hughes 研制的。

使用 mSQL 的方法很容易。首先，在站点 www.hughes.com.au 注册一个用户，就能免费下载该软件的最新版本；其次是安装，mSQL 的安装相当简单，二进制版本的安装只是将文件展开到用户某一目录下配置几个参数就可以了。一般 mSQL 数据库系统的所有文件都安装到/usr/local/Hughes 目录下。配置 mSQL 数据库所要做的工作就是修改/usr/local/Hughes 目录下的文件 msql.conf。

为了将 mSQL 集成到用户的应用程序中，mSQL 数据库系统为用户提供了一个 C 语言的 API 库和一个解释器，用户可以将 mSQL 集成到用标准 C 语言开发的应用程序中。而有了这个工具，用户可以通过编写类 C 语言脚本语言将 mSQL 数据库嵌入 html 文件中。

MySQL 是由瑞典的 T.c.X 公司开发的。该公司要使用内部数据库，且认为标准的 SQL 语言比较合适，于是将自己的 API 搭建在 mSQL 上，组成了 MySQL。用户可在 http://www.MySQL.com 上找到它。MySQL 是一个快速的、多线程、多用户和健壮的关系数

据库。MySQL 服务器支持关键任务、重负载生产系统的使用，也可以将它嵌入软件中。MySQL 不仅速度快，而且还有以下优点：开源；可在多处理器环境下使用；支持 C、C++、Java、Perl、PHP、Python 和 TCL API 等编程；支持多种操作系统；对于中小型，甚至大型应用它都能胜任工作，常与 PHP 等搭档，用于网站开发。

PostgreSQL 起源于由加利福尼亚开发的 Ingres 数据库，当时称为 Postgres。刚开始，Postgres 并不支持 SQL，大约在 1995 年，Jolly Chen 和 Andrew Yu 使之能够支持 SQL，并发布了一个新版本 Postgres95。1996 年加入开放源代码协会，并更名为 PostgreSQL。由于它没有与商业绑定，所以在许多 Linux 发行版本中都有免费的 PostgreSQL 版本。

12.8.7 PHP 开发

PHP（Personal Home Page）是一种跨平台的在服务器端执行的脚本语言，可以在 Windows、UNIX/Linux 系统上运行，尤其是与全球网站服务器普及和使用率最高的 Apache 整合后，速度大大提高，也可以为 CGI 使用。

PHP 是免费软件，用户可从网站 http://www.php.net 中得到所需的版本，也可以从 Linux 系统发行版本安装所需要的内容。PHP 可支持相当多的数据库，比如 dBase、Empress、Informix、InterBase、mSQL、MySQL、Oracle、PostgreSQL、Sybase 和 UNIX dbm 等。

由于免费和自由软件的原因，现在较流行的网站开发模式是 Apache+ MySQL+PHP，因此需要 Apache 服务器和 MySQL 服务的支持。如果是在 Linux 环境下使用 Apache+MySQL+PHP 开发模式，那就是 Linux+Apache+MySQL+PHP，即所谓的 LAMP，这是 Linux 环境网站建设与运行的普遍模式。

12.8.8 Objective-C 开发

Objective-C 通常简记为 ObjC、Obj-C 或 OC，是扩充 C 的面向对象编程语言。它主要用于 Mac OS X 和 GNUstep 这两个使用 OpenStep 标准的系统。Linux 支持 Objective-C 开发，但需要安装相关的软件。

1．安装 gnustep*及相关软件

```
#yum install gnustep* gcc-objc gcc-objc++                #红帽系列系统
#apt install gnustep*                                     #ubuntu 系统
```

2．hello world

设有 OC 版的 hello world 程序，文件名为 hello.m，清单如下所示。

```
#import <Foundation/Foundation.h>
int main(int argc, const char *argv[]) {
        NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
        NSLog(@"Hello OC world!\n");
          [pool drain];
        return 0;
}
```

3．编译

hello.m 编译和链接需要很多与 OC 相关的环境和库选择与设定，常见方法如下：

```
$ gcc `gnustep-config --objc-flags` -lobjc -lgnustep-base hello.m -o hello
$ gcc hello.m -o hello -lgnustep-base -lobjc `gnustep-config --objc-flags`
$ gcc hello.m $(gnustep-config --objc-flags) -lobjc -lgnustep-base -o hello
```

在 ubuntu 系统下要注意选项的先后次序，将源程序文件写在其他选项的前面。

12.8.9　Python

Python 是由荷兰人 Guido van Rossum 发明的，第一个公开发行版发行于 1991 年。Python 是从 ABC 语言发展起来的，主要受到 Modula-3 的影响，并且结合了 UNIX shell 和 C 的习惯。Python 是完全自由的，一切遵循 GPL。

Python 的设计哲学是“优雅”、“明确”、“简单”。在设计上坚持清晰划一的风格，这使得 Python 成为一门易读、易维护，并且被大量用户所欢迎、用途广泛的语言。它的应用涉及操作系统运维、游戏、图像、图表、科学计算、人工智能、XML、大数据、云计算、机器人、网络编程与 Web 开发等。由于 Python 语言的简洁性、易读性以及可扩展性，在国外用 Python 做科学计算的研究机构日益增多，一些知名大学已经采用 Python 作为程序设计课程。

Python 有以下特点：脚本语言，也是解释语言，强大的动态数据类型支持；兼容性好、跨平台、可移植；自动内存回收；面向对象特性；强大的类库支持，编程容易；易于扩展，便于拼接（嵌入）；开源、免费。

还有，单行语句和单行命令行输入（每行只书写一条语句），缩进成为语法的一部分。Python 使用缩进表示语句块的开始和结束，而不是像其他语言那样使用花括号或者其他定界符。这与其他语言有明显的不同，需要初学者注意，一不小心就会造成编译或语法错误，尤其要注意 Tab 与空格有区别。

Python 在执行时，首先将源代码编译成 Python 的字节码，然后再由 Python 虚拟机来执行。除此之外，Python 还可以交互模式运行，比如，可在命令模式下直接运行 Python 交互环境，可在交互环境输入操作指令或 Python 语句，让 Python 立即执行。idle 为 Python 的集成工作环境。

当用户在提示下输入 python 并回车时，将进入交互界面，以下为一个交互过程的记录。

```
>>>                                          #>>>为 Python 的提示符
>>>print "Hello Python World!\n"             #print 语句，要求输出"Hello Python World!\n"
>>> x,y,z=3,4,5                              #定义变量 x,y,z
>>> x**2,y**2,z**2                           #交互式输出 x**2,y**2,z**2 的值
>>> import math                              #导入数学库
>>> math.sqrt(x**2+y**2)                     #交互式输出 x**2+y**2 平方根
>>> z=3+4j                                   #定义一个复数变量 z
>>> z.real,z.imag                            #交互式分别输出 z 的实数和虚数部分
>>> math.sqrt(z.real**2+z.imag**2)           #交互式计算并输出 z 的模
>>> print("z=%d+%dj\n" %(z.real,z.imag))     #print 语句，按格式要求输出复数 z 的值
```

```
>>> print("The modulus of z is %d\n" %(modulus(z)))     #输出复数 z 的模
>>>                                                     #按^D 可结束交互
```

简化一下上述交互的过程，去除交互输出，再为其增加一个函数 modulus(z)，用于计算复数 z 模的值。再按照脚本文件的要求，正确地写好第一行，设脚本程序名为 hello.py，代码如下所示。

```
#!/usr/bin/python                                    #第一行
import math                                          #导入数学库
def modulus(z):                                      #定义函数
    r=math.sqrt((z*z.conjugate()).real)              #函数体（注意行首缩进）
    return r
                                                     #主程序开始（空行不必须）
print "Hello Python World!"                          #print 语句，要求输出"Hello Python World!\n"
x,y,z=3,4,5                                          #定义变量 x,y,z
z=3+4j                                               #定义变量 z
print("z=%d+%dj" %(z.real,z.imag))                   #显示复数 z
print("The modulus of z is %d" %(modulus(z)))        #输出 z 的模（调用了函数 modulus(z)）
```

为其增加执行权后，就可以直接执行了，方法如下：

```
$ chmod +x hello.py
$ ./hello.py
```

需要提醒的是，若第一行设置不正确，就只能使用 Python 直接解释执行了，方法如下：

```
$ python hello.py
```

12.8.10　R 语言

R 是用于统计分析、绘图的语言和操作环境。R 是属于GNU系统的一个自由、免费、源代码开放的软件，它是一个用于统计计算和统计制图的优秀工具，有 UNIX、Linux、MacOS 和 Windows 等版本，均可以免费下载和使用。在 R 的安装程序中只包含了少数几个基础或核心模块，其他外在模块可以通过 CRAN 获得。

1．R 语言的安装

R 语言的安装方法如下：

```
#yum install R-core R-devel                          #红帽系统
#apt-get install r-base                              #ubuntu
```

R 语言安装完成后，就可以在命令提示符下输入：

```
$ R
```

进入 R 语言环境了。“>”为 R 的命令提示符，用户可以输入 demo()查看演示清单，可以从清单选择一个题目，比如 error.catching，执行 demo(error.catching)来查看演示代码；还可以用 help()打开帮助界面，通过 help(topic)或 help(package=PACKAGE)对相关的主题或安装包进行

帮助；输入 help.start()可以打开一个 Web 界面的帮助工具，可以从界面上选择帮助主题。最后，用户可输入 q()或^D 退出 R 语言环境。

2．简单操作

在 R 环境中可以交互式地进行 R 语言相关操作，比如：

```
> print("Hello R World!")          #输出"Hello R World!" （忽略所有输出）
> v1 <- c(1,2,3,4)                 #定义向量变量 v1
> v1                               #交互式输出 v1
> v2<-2*v1                         #定义向量变量 v2：v2=2*v1
> v2                               #交互式输出 v2
> T<-table(v1,v2)                  #由 v1 和 v2 定义表 T
> T                                #交互式输出 T
> print(v1)                        #调用 print 输出 v1
> print(T)                         #调用 print 输出 T（结果略）
> print(T,zero.print='.')          #调用 print，以另一种方式输出 T（结果略）
>                                  #R 的提示符
```

将其中的交互输出行及所有的结果去除，并加上正确的第一行，以上内容变为如下的 R 代码。

```
#!/usr/bin/R -f
print("Hello R World!")
v1 <- c(1,2,3,4)
v2 <- 2*v1
T <- table(v1,v2)
print(v1)
print(T)
print(T,zero.print='.')
```

若将其保存为文件 hello.R，则在 R 交换环境下可以通过命令

```
> source(hello.R)
```

运行。若为 hello.R 增加执行权后，则可在命令行直接像执行命令一样执行它。方法如下：

```
$ chmod +x hello.R
$ ./hello.R
```

若第一行设置不正确的话，可用 R 命令直接解释执行它，方法是：

```
$ R hello.R
```

3．简单管理

R 语言的使用很大程度上是借助各种各样 R 包的辅助，这些 R 包就是针对 R 的插件，不同的插件满足不同的功能，例如，用于经济计量、财经分析、人文科学研究以及人工智能等。

1）包安装

首先输入命令 R，进入 R 语言环境，然后执行命令：

```
> install.packages("packagenames","dir")
```

packagenames 为单个包名，或由单个包名组成的向量（见示例），dir 为安装位置，不需指定，由安装程序自己确定。

由于我们没有指定足够的信息，在安装过程可能会从弹出的 HTTPS CRAN 中选择一个包源，比如"China（Beijing)[https]"，选择后，会自动安装。若没有指定包名则会在选择包源之后提示用户选择包名。比如，安装随机森林包（randomForest）和 CRAT 决策树包（tree）可以分别使用：

```
> install.packages("randomForest")
> install.packages("tree")
```

或

```
> install.packages(c("randomForest","tree"))
```

对于已经下载的包（*.tar.gz 文件）可以在本地进行安装。

2）加载包

包安装后，如果要使用它，则必须先加载它（在默认情况下，R 启动时默认会加载基本包），加载包命令为 library("package_name")或 require("package_name")，比如：

```
> library("randomForest")
> require("randomForest")
```

3）查看包的相关信息

（1）查看包帮助信息。

查看包帮助信息的方法有：

```
> library()                          #查看所有安装包
> library(help="package_name")       #比如 library(help="randomForest")
> help(package="package_name")       #比如 library(help="tree")
```

输出的主要内容包括：包名、版本、作者、更新时间、功能描述、开源协议等。

（2）查看已经加载的包。

查看当前环境已经加载的包，可使用命令：

```
> find.package()
> path.package()
```

（3）从内存中卸载包。

若要将某个已经加载的包从内存中卸载，可使用命令 detach()，detach()用法为：

```
detach(package:packagename)或 detach(编号)
```

其中，packagename 为包名，“编号”为欲卸载包在 find.package()输出中的编号，比如：

```
> detach(package: randomForest)
> detach(2)                          #假设 2 为欲卸载包在 find.package()输出中的编号
```

（4）加载包中的数据集。

```
data(dsname, package="package_name")
```

比如：

```
> data()                                    #列出所有可用数据集
> data(package = "rpart"))                  #列出 rpart 包中的数据集
> data(USArrests, "VADeaths")               #加载'USArrests'和'VADeaths'中的数据集
```

12.8.11　虚拟化、云计算和大数据应用

虚拟化、云计算和大数据是当今计算机或 IT 界最响亮的几个词，或者说我们周围总是充斥着这些词。对于大数据和云计算有多种不同的描述，它们有区别又有联系，很难说准确。用一句直白话可描述为，云计算是硬件资源的虚拟化；大数据是海量数据的高效处理，但都离不开虚拟化。

开源的 Openstack 和 Hadoop 是当前比较有活力的云计算和大数据平台，这两者在 Linux 下都有很好的表现。

1．虚拟化

计算机中的虚拟化（Virtualization）是一种资源管理技术，是指将计算机的各种实体资源（如服务器、网络、内存及存储等）予以抽象、转换后呈现出来，打破实体结构间的不可分隔的障碍，不再受现有资源的架构方式、地域或空间的限制，可以使用户以比原本的组态更好的方式来应用这些资源。一般所指的虚拟化资源包括计算能力和资料存储。Linux 提供 XEN 和 KVM 两种虚拟技术支持。

2．OpenStack

OpenStack 是一个开源项目，支持几乎所有类型的云环境，项目目标提供实施简单、可大规模扩展、丰富、标准统一的云计算管理平台。

OpenStack 覆盖了网络、虚拟化、操作系统、服务器等各个方面，也有很多版本。首个版本是 2010 年的 AUSTIN，较新版本是 2014 的 JUNO、2015 年的 KILO、LIBERTY 和 2016 年的 MITAKA，当前的版本为 Ocata，这些版本均可在 rhel、centOS 和 ubuntu 系统上使用，用户可以从 https://www.openstack.org/ 得到安装包，还可从 https://docs.openstack.org/ 或 https://docs.openstack.org/zh_CN/ 上获取安装部署文档。

3．Hadoop

Hadoop 是一个由 Apache 基金会开发的分布式系统基础架构，是一个开源项目。用户可以在不了解分布式底层细节的情况下开发分布式程序，充分利用集群的威力进行高速运算和存储。Hadoop 有高可靠性、高扩展性、高效性、高容错性和低成本等优点。

Hadoop 的框架最核心的设计就是 HDFS 和 MapReduce。HDFS 为海量的数据提供存储，MapReduce 为海量的数据提供计算。HDFS 就像一个传统的分级文件系统，可以实现对文件的创建、复制、删除、更名等操作。存储在 HDFS 中的文件被分成块，然后将这些块复制到

多个计算机中（DataNode），实现分布式存储与访问。

Hadoop 的系统软件可从 http://www.hadoop.net/dyn/closer.cgi 建议的站点下载所需的 Hadoop 软件包，Hadoop 的当前版本是 3.0.0，当前的稳定版本是 2.7.3，所对应的软件包为 hadoop-2.7.3.tar.gz 和 hadoop-2.7.3-src.tar.gz。用户可从 http://hadoop.apache.org/docs/得到相应版本的安装配置文档。

习题 12

1．思考题

（1）gcc 是怎么工作的？
（2）如何构造和使用自己的静态库？
（3）如何构造和使用自己的共享库？
（4）在 gdb 调试程序的过程中，如何确定用户所在的模块？

2．填空题

（1）设有 C 语言程序 my.c
A．生成目标文件 my.o 的命令是（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）。
B．生成汇编语言文件 my.s 的命令是（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）。
C．生成可执行程序 myp 的命令是（＿＿＿＿＿＿＿＿＿＿＿＿＿）。
（2）设有两个 C 语言程序模块 c1.c 和 c2.c（不含 main 函数）
A．由 c1.c 和 c2.c 生成静态库 libmyar.a 的命令是（＿＿＿＿＿＿＿＿＿＿）。
B．由 c1.c 和 c2.c 生成共享库 libmyar.so 的命令是（＿＿＿＿＿＿＿＿＿＿）。
（3）在某 Makefile 内有以下语句：

```
SRC = f1.c f2.c f3.c
TGT = $(SRC:.c=.d)
SRC += f4.c
TGT=____________________
SRC=____________________
```

请完成填空。

3．综合题

阅读 Makefile 文件：

```
all : libmys.so
SRC = f1.c f2.c f3.c
TGT = $(SRC:.c=.o)
%.o : %.c
	cc -fPIC -c $?
libmys.so : $(TGT)
	cc -shared -o $@ $(TGT)
```

```
clean:
	rm -f $(TGT)
```

回答以下问题：

（1）此 Makefile 文件的主要功能是什么？

（2）此 Makefile 文件包含多少个规则？它们分别是什么？

（3）使用此 Makefile 文件可以生成目标文件 f2.o 吗？为什么？

实验 12

1．对书中的多模块工程进行编译和执行。

2．静态库的构造与使用。

3．共享库的构造与使用。

网络服务与应用篇

- 网络时间服务 NTP
- 基础服务 OpenSSH 和 VNC
- Web 服务器 Apache
- DHCP 服务器
- FTP 与 TFTP 服务器
- 网络资源共享服务
- 域名服务器 DNS

第 13 章　网络时间服务 NTP

13.1　NTP 协议

13.1.1　NTP 简介

随着各种计算机网络应用的不断涌现，计算机的时间同步问题显得越来越重要了。为了避免时间错误，可从网络、时间源上获取时间，实现系统内各主机的时间同步。NTP（Network Time Protocol）是网络时间协议，是用来同步网络中各个计算机时间的协议。

NTP 提供准确时间，但先要有准确的时间来源，这一时间是国际标准时间 UTC。NTP 获得 UTC 的时间来源，可以是原子钟、天文台、卫星，也可从 Internet 上获取。每台计算机都安装时间接收设备是不现实的，作为替代，指定作为时间服务器的计算机从时间源接收时间，并使用 NTP 协议来同步网络内计算机系统的时间。

NTP 所建立起的网络基本结构是分层管理的类树状结构。网络中的节点有两种可能：时钟源或客户端。每一层的时钟源或客户可向上一层或本层的时钟源请求时间校正。在此结构中，时间按 NTP服务器的等级传播。按照离外部 UTC 源的远近程度将所有服务器归入不同的 Stratum（层）。Stratum-1 在顶层，由外部 UTC 接入，而 Stratum-2 则从 Stratum-1 获取时间，Stratum-3 从 Stratum-2 获取时间，依次类推，但 Stratum 层的总数限制在 15 以内。最外层提供 UTC 时间源，为 Stratum-0，但不在 NTP 层次之内。也允许使用 Stratum-16，表明该层距时间源非常远，不能提供可靠的时间。

计算机主机一般同多个时间服务器连接，利用统计学的算法过滤来自不同服务器的时间，以选择最佳的路径和来源来校正时间。即使主机在长时间无法与某一时间服务器相联系的情况下，NTP 服务仍可有效运行。

为防止对时间服务器的恶意破坏，NTP 使用了识别（Authentication） 机制。

在通常情况下，NTP 客户端发出时间请求，与时间服务器交换时间，这个交换的结果是，从服务器得到标准时间，且客户端能计算出时间的延迟，并基于此两者调整本地系统时间。冗余服务器和不同的网络路径可用于保证可靠性的精确度。除了客户端/服务器同步以外，NTP 还支持同等计算机的广播同步。

13.1.2　NTP 的工作模式

主动/被动对称模式：一对一的连接，双方均可同步对方或被对方同步，先发出申请建立连接的一方工作在主动模式下，另一方工作在被动模式下。此方式适用于配置冗余的时间服务器，可以提供更高的精确度给主机。

客户/服务器模式：与主/被动模式基本相同。唯一区别在于，客户方可被服务器同步，但服务器不能被客户同步。

广播/组播模式：一对多的连接，服务器不论客户工作在何种模式下，主动发出时间信息，

客户由此信息调整自己的时间。但此方式的精确度不高（仅秒级），对时间精确度要求不是很高的情况下可以采用。

每一个时间包内包含最近一次事件的时间信息、包括上次事件的发送与接收时间、传递现在事件的当地时间及此包的接收时间。在收到上述包后即可计算出时间的偏差量与传递信息的时间延迟。时间服务器利用过滤算法及先前八个校时资料计算出时间参考值，判断后续校时包的精确性。仅从一个时间服务器获得校时信息，不能校正通信过程所造成的时间偏差，若同时与多个时间服务器相关联，就可利用过滤算法从中找出相对较可靠的时间源，然后采用它来校正时间。

13.1.3　NTP服务器选择

现在有很多时间服务器可以使用，比如本书所用的系统，都提供自己的时间服务器地址，可在安装 ntp 软件包之后查看 NTP 服务配置文件/etc/ntp.conf 的内容。国内常用的 NTP 服务器有 time.buptnet.edu.cn、s1a.time.edu.cn、s2b.time.edu.cn 和 time.jmu.edu.cn 等，在 Windows 系统中提供的有 time.nist.gov、time-a.nist.gov、time-b.nist.gov 和 time.windows.com 等。用户也可以建立自己的时间服务器。

13.2　通过 ntp 配置 NTP

13.2.1　软件包的安装

NTP 服务软件包为 ntp，相关的软件包还有 ntp-doc 和 ntpdate 等，可根据需要有选择地或全部安装它们，方法如下：

```
#yum install ntp ntp-doc ntpdate                #红帽系统
#apt install ntp ntp-doc ntpdate ntpstat        #Ubuntu
```

13.2.2　服务管理

NTP 的服务名在红帽和 Ubuntu 下分别为 ntpd 和 ntp，可用服务管理的方法管理它们。

1）红帽系统

```
#systemctl status ntpd                  #检查运行 ntpd 状态
#systemctl enable/disable ntpd          #启用/禁用 ntpd
#systemctl  start/stop/restart ntpd     #启动/停止/重启 ntpd
```

2）Ubuntu 系统

```
#systemctl status ntp                   #检查运行 ntp 状态
#systemctl enable/disable ntp           #启用/禁用 ntp
#systemctl  start/stop/restart ntp      #启动/停止/重启 ntp
```

需要提醒的是，每当修改配置文件后，都需要重启 NTP 服务。

13.2.3　配置文件及配置

1．配置文件 ntp.conf 及默认配置

ntpd 的配置文件为/etc/ntp.conf，通过命令：

```
#sed -e '/^[[:blank:]]*#.*$/d;/^[[:blank:]]*$/d' /etc/ntp.conf
```

可以得到其中默认的有效配置项（如表 13-1 所示）。

表 13-1　/etc/ntp.conf 中的默认配置

Fedora 24	Ubuntu
	driftfile /var/lib/ntp/ntp.drift
	statistics loopstats peerstats clockstats
driftfile /var/lib/ntp/drift	filegen loopstats file loopstats type day enable
restrict default nomodify notrap nopeer noquery	filegen peerstats file peerstats type day enable
restrict 127.0.0.1	filegen clockstats file clockstats type day enable
restrict ::1	pool 0.ubuntu.pool.ntp.org iburst
server 0.fedora.pool.ntp.org iburst	pool 1.ubuntu.pool.ntp.org iburst
server 1.fedora.pool.ntp.org iburst	pool 2.ubuntu.pool.ntp.org iburst
server 2.fedora.pool.ntp.org iburst	pool 3.ubuntu.pool.ntp.org iburst
server 3.fedora.pool.ntp.org iburst	pool ntp.ubuntu.com
includefile /etc/ntp/crypto/pw	restrict -4 default kod notrap nomodify nopeer noquery limited
keys /etc/ntp/keys	restrict -6 default kod notrap nomodify nopeer noquery limited
disable monitor	restrict 127.0.0.1
	restrict ::1
	restrict source notrap nomodify noquery

在表 13-1 中，红帽系统只选择了一个代表：Fedora 24，与红帽其他系统的差别只在服务设置项上，比如第一个 server 所在行的内容分别为 server 0.fedora.pool.ntp.org iburst、server 0.centOS.pool.ntp.org iburst 和 server 0.rhel.pool.ntp.org iburst，再比较一下 Ubuntu 的 pool 0.ubuntu.pool.ntp.org iburst 可以看出，它们各自有自己的 NTP 服务器。

2．/etc/ntp.conf 中常见配置项

文件/etc/ntp.conf 中的配置项有访问控制、认证、运行监督、时钟引用和杂项等配置内容，简述如下。

1）访问控制选项

discard [average avg] [minimum min] [monitor prob]：设置速率控制参数，保护服务器不受客户滥用。如果在 ACL 中存在限制标志，则违反这些限制的数据包将被丢弃。如果添加了 kod 限制，则返回一个 kiss-o'-death 包。average avg、minimum min 和 monitor prob 分别用于指定包间最小平均间隔时间、最小时间间隔和溢出速率控制窗口的数据包丢弃的概率。

restrict address [mask mask] [flag] [...]：约束定义。其中 address 可以是 IP，也可以是 DNS 能够解析的主机名，但必须在 ntpd 启动时能够被解析，若解析出多个 IP 地址，则只有首个

被使用。一个 default（IP 0.0.0.0，mask 0.0.0.0）项总是要包括进来，尽管没有配置它。一个没有 flag 的 default 项将不进行任何约束或限制。常用的 flag 如下所述。

- ignore：拒绝所有包，包括 ntpq 和 ntpdc 查询，相当于关闭所有的 NTP 联机服务。
- kod：如果一个包违反了 discard 建立的限制率，且有 limited 标志存在，则发送一个 kiss-o'-death(KoD)包。
- limited：如果违反 discard 建立的限制率，则拒绝服务，但对 ntpq 和 ntpdc 查询无效。
- nomodify：拒绝 ntpq 和 ntpdc 改变服务状态，但时间服务正常开展。
- noquery：拒绝 ntpq 和 ntpdc 查询，但时间服务正常开展。
- nopeer：拒绝所有非经过验证的可能引起联合的包。
- noserve：拒绝除 ntpq 和 ntpdc 以外的请求包。
- notrap：拒绝提供模式 6 控制消息 trap 服务来匹配主机。trap 服务是 ntpdc 用于远程事件日志消息控制协议的子系统。
- notrust：拒绝接收没有加密验证的信息包。

2）配置命令

server address [options …]：用于 C/S 模式，指定服务器地址及选项。address 可以是 IP 或 DNS 主机名；如果有 prefer 选项，则将该 server 设为首选；如果指定了 preempt（抢占）标志，则会启动抢占式客户端模式关联。

pool address [options …]：类似 server 模式。用于实现池自动服务器发现方案的 C/S 模式关联。

peer address [options …]：用于主动/被动对称模式，指定远程对端服务器地址及选项。

broadcast address [options …]：用于 broadcast 或 multicast 服务器模式，指定广播（broadcast）或多播（multicast）地址及选项。

manycastclient address [options …]：用于 manycast 客户端模式，指定多播（multicast）组地址。

常用的配置命令选项如下所述。

- burst：当服务器可到达时，发送 8 个数据包，而不是通常的一个，只适用于 server 类类型。
- iburst：当服务器不可到达时，发送 8 个数据包，而不是通常的一个，只适用于 server 类型。
- preempt：将关联指定为抢占型，而不是默认的持久化。该选项会被广播模式忽略，但对 many castclient 和 pool 模式有用。
- prefer：设为首选。
- autokey：发送和接收通过自动密钥方案认证的数据包，与 key 选项互斥。
- key N：指定密码文件中的第 N 条被使用。

3）其他和杂项配置

driftfile file：指定 driftfile（漂移文件）的位置及名字。漂移文件用于存储本地系统时钟运行时的频率偏移量。如果存在，则在系统启动时读取它，并用于纠正时钟源。

includefile includefile：指定包含文件。

keys file：指定密码文件。

keysdir pathK：指定自动密钥认证方案密码文件路径，默认为/etc/ntp/crypto。

enable/disable [auth|bclient|calibrate|kernel|monitor|ntp|pps|stats]：启用或禁用 auth、bclient、calibrate、kernel、monitor、ntp、pps 和 stats 等选项。

13.2.4　文件/etc/sysconfig/ntpd 和/etc/default/ntp

文件/etc/sysconfig/ntpd 和/etc/default/ntp 的内容分别为红帽系统和 Ubuntu 系统的 NTP 服务器守候进程 ntpd 的默认启动参数，其内容为：

```
OPTIONS="-g"                    #红帽系统
NTPD_OPTS='-g'                  #Ubuntu 系统
```

-g 的意义可参见命令的 ntpd man 手册。

13.2.5　防火墙设置

若要让 NTP 服务器能够工作，还必须在防火墙规则设置上开放对 ntp 服务的限制，方法如下。

```
#firewall-cmd --add-service ntp       #firawalld 防火墙
#ufw allow ntp                        #ufw 防火墙
```

13.2.6　配置文件实例

任务：在局域网 192.168.137.0 内配置 NTP 服务器，并实现网内的系统时间同步。

设 NTP 服务器的地址为 192.168.137.10，使用外部服务器 time.nist.gov 作为 UTC 时间源。

1）服务器

（1）红帽系统

编辑配置文件/etc/ntp.conf，在行

```
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

下添加如下一行内容（事实上是参照上一行修改而得到的）：

```
restrict 192.168.137.0 mask 255.255.255.0 nomodify notrap
```

意为允许 192.168.137.0/24 内的主机访问 NTP 服务器，但有 nomodify 和 notrap 限制。

在新添加行之下，第一个“server …”开头的行之前，添加服务器设置项如下：

```
server time.nist.gov iburst
```

之后，可以注释掉其他的 server 或 pool 所在的行，也可以不注释掉。配置文件的其他地方保持不变。

此时经修改后的配置文件/etc/ntp.conf 的局部如下所示：

```
……
restrict 127.0.0.1
restrict ::1
```

```
#Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
restrict 192.168.137.0 mask 255.255.255.0 nomodify notrap

#Use public servers from the pool.ntp.org project.
#Please consider joining the pool (http://www.pool.ntp.org/join.html).
server time.nist.gov iburst
server 0.fedora.pool.ntp.org iburst
server 1.fedora.pool.ntp.org iburst
……
```

（2）Ubuntu 系统

对于 Ubuntu 系统，处理方法与红帽系列基本相同。编辑配置文件/etc/ntp.conf，在行

```
#restrict 192.168.123.0 mask 255.255.255.0 notrust
```

下添加如下一行内容：

```
restrict 192.168.137.0 mask 255.255.255.0 nomodify notrap
```

意为允许 192.168.137.0/24 内的主机访问 NTP 服务器，但有 nomodify 和 notrap 限制。

第一个“pool …”开头的行之前，添加服务器设置项如下：

```
server time.nist.gov iburst
```

之后，可以注释掉其他的 server 或 pool 所在的行，也可以不注释掉。

配置完成后，使用以下命令重启 NTP 服务器：

```
#systemctl restart ntpd                 #红帽系统
#systemctl restart ntp                  #Ubuntu
```

2）客户端

对 NTP 客户端的配置，所要做的工作要少得多，只需要添加一个 server 配置项就可以了。因为 NTP 服务器的 IP 为 192.168.137.10，故需要按服务器的配置方法添加如下的配置行：

```
server 192.168.137.10 iburst
```

配置完成后，重启 NTP 服务器。

13.2.7　配置测试

1）检查 ntp 服务端口的使用情况

ntp 服务所使用的端口号为 123（可能过命令 grep '^ntp' /etc/services 来查询），使用 netstat 命令可以查询该端口是否处于监听状态，命令如下：

```
netstat -unl | grep 123        #查看 123 端口，输出略
```

2）ntpstat

ntpstat 用于检查系统与服务器的时间同步状态。NTP 服务器配置完成并重启后，并不会马上就进行时间同步，可用 ntpstat 来检查是否已经同步。如果已经同步，还可报告时

间精度。

```
#ntpstat                  #未同步时
  unsynchronised
      polling server every 8 s
#ntpstat                  #同步后
synchronised to unspecified at stratum 2
      time correct to within 1141 ms
      polling server every 64 s
#ntpstat                  #服务器未运行
Unable to talk to NTP daemon. Is it running?
```

3）使用 ntpq 监控 NTP 时间服务器

ntpq 可用于监视 NTP 守护进程 ntpd 工作并确定性其能，它可应用于命令行模式或交互模式。ntpq 的用法为：

```
ntpq [-46dinp] [-c command] [host] […]
```

选项-4/-6 用于 IPv4/IPv6 主机或地址模式。-i：交互模式；-d：debug 模式；-n：输出中的主机为 IP 形式；-c command：执行 ntpq 子命令；-p：输出已知的 peer 列表以及其状态的摘要，等同于-c peers。

当无参数和选项执行 ntpq 时进入交互模式，在此模式下可以通过 help 获得帮助。

（1）ntpq -p 或 ntpq -c peers

ntpq -p 或 ntpq -c peers 用于查询网络中的 NTP 服务器，同时显示客户端和每个服务器的关系。

```
#ntpq -p
     remote        refid        st t when poll reach    delay     offset    jitter
==============================================================================
  time-c.nist.gov    .NIST.      1 u   36   64     1   253.544   −58.279    0.000
```

remote：响应这个请求的 NTP 服务器的名称；refid：NTP 服务器 id；st：正在响应请求的 NTP 服务器的级别；t：为类型：u-单播，b-广播，l-本地；when：上一次成功请求之后到现在的秒数；poll：当前请求的时钟间隔的秒数；reach：八进制数，表示最近 8 次时钟同步包接收情况（每接收一个包左移一位，非 0 表示成功，0 表示失败）；delay：客户端从发出请求到接收到回复所经过的时间（毫秒）；offset：NTP 客户端本地时间与 NTP 服务器时间的差异（毫秒），也就是客户端需要调整的时间差异；jitter：系统时间与 BIOS 时间的差。

这是注释掉其他服务器，只保留了一个 time.nist.gov 的情况，故输出中只有所配置的服务器一行。若有多个时间服务器，则每个都会有一行输出，输出的行首会有一些标志。“*”：正在使用的 NTP 服务器；“+”：可连上线，而且可作为候选；“空格/X/./-”：因为某种原因被丢弃。

（2）ntpq -c assoc

ntpq -c assoc 用于获取并显示所关联服务器的标识和状态列表。输出时，按照配置文件/etc/ntp.conf 中的时间服务器先后顺序依次显示各服务器的状态。

```
ind   assid   status   conf   reach auth   condition   last_event   cnt
```

```
=====================================================
  1   9705   961a    yes    yes   none   sys.peer      sys_peer   1
  2   9706   932a    yes    yes   none    outlyer      sys_peer   2
  3   9707   941a    yes    yes   none candidate      sys_peer   1
  4   9708   8011    yes     no   none     reject      mobilize    1
  5   9709   942a    yes    yes   none candidate      sys_peer   2
```

各字段的意义如下所述。

ind 索引号；assid：由服务器返回的关联标识；status:状态字；conf：是否配置；reach：是否可达；auth：认证情况，可取值有 ok、yes、bad 和 none；condition：reject/falsetick/excess/outlyer 对应 ntpq -p 输出中的“空格/X/./-”，candidate/backup/sys.peer 对应“+/#/*”；last_event：最后事件：mobilize（启动关联）/demobilize（解除关联），reachable（可达）/unreachable（不可达），restart（重启动），no_reply（无应答），access_denied（拒绝访问），sys_peer（变为正在使用的 NTP 服务器），clock_event（时钟事件），bad_auth（认证未通过）。

13.2.8　无 NTP 服务的 NTP 客户端

在配置有 NTP 服务的系统中，可以从网络时间服务器自动同步系统时间。Linux 提供有软件包 ntpdate，其中包含有同名的命令 ntpdate，使用它可在不安装和配置 NTP 服务器的系统中实现从 NTP 服务器同步系统时间。ntpdate 用于时间同步的简单用法为：

```
ntpdate NTP_server
```

比如，从 time.nist.gov 和 192.168.137.10 同步时间的命令分别如下：

```
#ntpdate time.nist.gov              #从 time.nist.gov 同步系统时间
#ntpdate 192.168.137.10             #192.168.137.10 同步系统时间
```

需要提醒的是，当刚开始使用时可能会出现“… no server suitable for synchronization found”的提示。

经过同步之后，可以使用 date 命令来查询时间，但这个系统时间，并不是硬件时间，在下次启动时，可能会恢复为硬件时间。若要更改系统硬件时间，可以使用 hwclock -w 命令，将系统时间同步到硬件时钟，方法是：

```
#hwclock -w                         #将系统时间同步到硬件时钟
```

如果要在不安装和配置 NTP 服务的情况下保持系统和硬件时钟的正确性，可使用 cron 机制周期性地校准时间。比如，若让系统在每天的 5:30 从 NTP 服务器 192.168.137.10 进行时间校准，并同时更新硬件时间，可以 root 用户身份运行 crontab -e 命令，向 crontab 文件增加如下一行内容：

```
30 5 * * * /usr/sbin/ntpdate 192.168.137.10 &&/sbin/hwclock -w
```

或直接在/etc/crontab 中添加一行如下内容：

```
30 5 * * * root /usr/sbin/ntpdate 192.168.137.10 && /sbin/hwclock -w
```

13.3 通过 chrony 配置 NTP

chrony 是一个开源的自由软件，用于保持系统时钟与 NTP 服务器的同步。它由 chronyd 和 chronyc 两个程序组成。chronyd 是后台守护进程，用于调整内核中运行的系统时钟和时钟服务器同步。chronyc 提供了一个用户界面，用于监控 chronyd 性能并进行多样化的参数配置，它可以工作在 chronyd 实例控制的计算机上，也可以工作在远程计算机上。

chrony 可以同时作为 NTP 服务的客户端和服务端。

ntpd 和 chronyd 的主要区别之一是用于控制计算机时钟的算法。chronyd 在以下方面比 ntpd 有优势：

- 当外部时间引用只是间歇性地可访问时，chronyd 可以很好地工作，而 ntpd 需要定期轮询时间引用才能工作正常。
- 即使在网络长时间被阻塞时，chronyd 也能表现很好。
- chronyd 可以使时钟更快地同步，并具有更好的时间精度。
- chronyd 能够快速适应，比如由于温度变化引起的晶振频率突变，但 ntpd 可能需要很长时间才能稳定下来。
- 在默认配置中，当时钟在系统启动时已经同步时，chronyd 将不会步进（steps the time）时间，以免打乱其他运行程序。ntpd 也可以被配置为永不步进时间，但是它必须使用不同的方法来调整时钟。
- chronyd 可以在一个更大范围内调整时钟频率，这使得它可以在有缺陷或不稳定时钟的机器上运行，比如，在一些虚拟机上。

关于 chrony 的更多信息可参考 https://chrony.tuxfamily.org/或《Fedora 24 System Administrator's Guide》以及相关的在线文档。

13.3.1 chrony 软件包的安装

与 chrony 服务器相关的软件包只有 chrony 一个，在红帽系统和 Ubuntu 下也是相同的，安装命令如下。

```
#yum install -y chrony                 #红帽系统
#apt install -y chrony                 #Ubuntu 系统
```

13.3.2 服务管理

chrony 服务器的服务名在红帽系统和 Ubuntu 下分别为 chronyd 和 chrony，可以使用服务管理工具管理它。

1. 红帽系统

```
#systemctl status chronyd                  #状态检查
#systemctl enable/disable chronyd          #启用/禁用
#systemctl start/stop/restart chronyd      #启动/停止/重启服务
```

2．Ubuntu 系统

```
#systemctl status chrony                    #状态检查
#systemctl enable/disable chrony            #启用/禁用
#systemctl start/stop/restart chrony        #启动/停止/重启服务
```

需要提醒的是，为使修改后的配置文件生效，需要重启服务。

13.3.3 配置文件 chrony.conf 及默认配置

chrony 服务使用的配置文件均为 chrony.conf，在红帽和 Ubuntu 下的位置分别为/etc/和/etc/chrony/，配置内容及格式和 ntp.conf 相似。可使用

```
#sed -e '/^[[:blank:]]*#.*$/d;/^[[:blank:]]*$/d' chrony.conf    #注意文件位置
```

得到其默认配置内容（见表 13-2）。

表 13-2 chrony.conf 的默认配置

红帽 fedora 24	Ubuntu
pool 2.fedora.pool.ntp.org iburst driftfile /var/lib/chrony/drift makestep 1.0 3 rtcsync logdir /var/log/chrony	pool 2.debian.pool.ntp.org offline iburst keyfile /etc/chrony/chrony.keys commandkey 1 driftfile /var/lib/chrony/chrony.drift log tracking measurements statistics logdir /var/log/chrony maxupdateskew 100.0 dumponexit dumpdir /var/lib/chrony logchange 0.5 hwclockfile /etc/adjtime rtcsync

chrony.conf 中的常见配置项见表 13-3。

表 13-3 chrony.conf 中的常见配置项

配 置 项	意 义
server/pool address	同 ntp
driftfile file	同 ntp
allow address	允许客户端通过 address 访问。address 可以为主机名、网段等
cmdallow/cmddeny address	允许/拒绝通过 address 使用 chronyc 对 chronyd 进行控制，address 意义同上。也可以是 cmddeny all，cmdallow all
rtcfile file	指定时钟精度跟踪参数存放文件
rtcsync	周期性地（Linux 每 11min）将系统时间复制到硬件时间，不能与 rtcfile 同时使用
dumdir	定义时间增益或损失测量历史存放目录

（续表）

配　置　项	意　　义
dumponexit	每当 chronyd 退出时，要保存每个时间源测量历史记录
local	本地时间源模式。允许 chronyd 服务器作为时间源。用于孤立网络中的主计算机。可用选项有 stratum L、distance D 和 orphan，如 local stratum 10 orphan
log measurements	将原始的 NTP 测量值和相关信息记录到 measurements.log 中
log statistics	将关于回归处理的信息记录到 statistics.log 中
log tracking	将系统的增益或损失率估计值的更改记录到 tracking.log 中
log rtc	记录系统的实时时钟 rtc
log refclocks	记录原始和过滤的参考时钟值到 refclocks.log 中
log tempcomp	将温度测量和系统速率补偿记录到 tempcomp.log 中
logdir dir	指定日志目录，如 logdir /var/log/chrony
logchange threshold	如果时钟调整大于 threshold 秒，则向 syslog 发送消息
keyfile file	指定密码文件为 file
commandkey N	指定密码文件中的第 *N* 条被使用，默认为 1
generatecommandkey	指定产生一个加密的密码，存放在 keyfile 指定的文件 file 中
hwclockfile file	告诉 chronyd 解析 file 文件，以了解实时时钟是否保持本地时间或 UTC。它覆盖 rtconutc 指令
makestep threshold limit	通常，chronyd 将根据需求通过减慢或加速时钟，使得系统逐步纠正所有时间偏差。makestep 强制 chronyd 在调整期大于某个阈值（threshold）时步进调整系统时钟，但只有在 chronyd 启动后未超过指定限制（limit）时才生效（可使用负值来禁用限制）

13.3.4　配置实例

任务:在局域网 192.168.137.0 内使用 chrony 配置 NTP 服务器,实现网内系统的时间同步。设服务器的地址为 192.168.137.10，使用外部服务器 time.nist.gov 作为 UTC 时间源。

1）服务器

（1）红帽系统（以 Fedora 24 为例）

编辑配置文件/etc/chrony.conf，在行

```
#allow 192.168/16
```

下，添加如下一行内容（事实上是参照上一行修改而到）：

```
allow 192.168/16        或
allow 192.168.137/24
```

意为允许 192.168/16 或 192.168.137/24 内的主机访问本服务器。

在行

```
pool 2.fedora.pool.ntp.org iburst
```

上，添加服务器配置项如下：

```
server time.nist.gov iburst
```

之后，可以注释掉其他的 server 或 pool 开始的行，也可以不注释掉。配置文件的其他地方保持不变。

此时，经修改后的配置文件/etc/ntp.conf 的局部内容如下所示：

```
… …
server time.nist.gov iburst
pool 2.fedora.pool.ntp.org iburst

… …

#Allow NTP client access from local network.
#allow 192.168/16
allow 192.168.137/24
```

（2）Ubuntu 系统

对于 Ubuntu 系统，处理方法与红帽系列相同。所不同的是，要把

```
server time.nist.gov iburst
```

添加到第一个 pool 开头的行之前。可注释掉 pool 开头的行，也可不注释。在行

```
#allow foo.example.net
```

之前添加

```
allow 192.168/16        或
allow 192.168.137/24
```

其他内容保持不变。配置完成后，重启 chrony 服务器：

```
#systemctl restart chronyd              #红帽系统
#systemctl restart chrony               #Ubuntu
```

2）客户端

对客户端的配置，只需要添加一个 server 配置项，并注释掉其他的 server 或 pool 项就可以了。因为服务器的 IP 为 192.168.137.10，故需要按服务器的配置方法添加如下的配置行：

```
server 192.168.137.10 iburst
```

配置完成后，重启 chrony 服务器。

13.3.5　配置测试

检查 chrony 实现的 NTP 服务器，可以使用客户端程序 chronyc，这里只出于对 chrony 服务器测试的目的，简单介绍 chronyc 的应用，关于 chronyc 对 chronyd 管理等由读者自行探索。chronyc 的用法为：

```
chronyc [OPTION] ... [COMMAND] ...
```

当不使用任何选项和参数运行 chronyc 时，将进入交互界面。在交互界面下可通过 help 命令得到帮助。

1．显示有关系统时钟性能的参数

```
#chronyc tracking
Reference ID     : 216.229.0.179 (nist1-lnk.binary.net)        #引用时间源 ID
Stratum          : 2                                           #层数为 2
Ref time (UTC)   : Sat Jun 10 08:32:24 2017                    #UTC 时间
System time      : 0.000039017 seconds slow of NTP time        #系统时间与 NTP 时间差
Last offset      : -0.000041500 seconds                        #最后校正时的时差
RMS offset       : 0.000579462 seconds                         #长期偏移的平均值
Frequency        : 8.058 ppm fast                              #如果不纠正的时钟错误率
Residual freq    : -0.007 ppm                                  #当前选择参考源的剩余率
Skew             : 1.028 ppm                                   #频率偏离值
Root delay       : 0.219461 seconds                            #最终同步的第 1 层延迟总和
Root dispersion  : 0.000668 seconds                            #最终同步的第 1 层累积偏差
Update interval  : 519.3 seconds                               #最后两次时钟更新间隔
Leap status      : Normal    #跳跃状态：Normal/Insert second/Delete second/Not synchronized
```

2．检查 NTP 时间源服务器状态

```
#chronyc sourcestats          #输出为注释掉其他 server 或 pool 的情况
#服务器端输出为（命令为 chronyc sourcestats）
210 Number of sources = 1
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
==============================================================================
nist1-lnk.binary.net        3   3     5   +171.606     261015   -270us    892us
客户端输出为（命令为 chronyc -n sourcestats）
210 Number of sources = 1
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
==============================================================================
192.168.137.24              0   0     0     +0.000   2000.000     +0ns   4000ms
```

3．检查时间源

```
#chronyc sources              #输出为注释掉其他 server 或 pool 的情况
#服务器端输出为（命令为 chronyc sources）
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* nist1-lnk.binary.net          1   6   377     0    +72us[ +159us] +/-  121ms
客户端输出为（命令为 chronyc -n sources）
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^? 192.168.137.24                0   9     0   10y    +0ns[   +0ns] +/-    0ns
```

若要得到更详尽的信息，可使用-v 选项，比如:

```
#chronyc sources -v
#chronyc -n sources –v
```

4．手工调整系统时间

```
#chronyc
chrony> password commandkey-password
200 OK
chrony> makestep
chrony>quit
```

13.3.6　其他说明

ntpdate 仍然可以与 chrony 配置的 NTP 服务器同步时间；ntp 服务可以与 chrony 服务器兼容，即在同一网络内，既可以 ntp 作为服务器，也可以 chrony 作为客户端，反之亦然。

习题 13

1．简述什么是 NTP。如何使用 NTP 协议同步网络时间？
2．什么是时区？中国的标准时区是什么？如何正确设置系统时区？
3．有几种办法或工具可以实现系统的时间管理，试说明如何使用这些工具管理系统时间。

实验 13

（1）在自己的局域网内，配置 ntp 服务器或客户端，实现网内系统的时间同步。
（2）在自己的局域网内，配置 chrony 服务器或客户端，实现网内系统的时间同步。

第 14 章　基础服务 OpenSSH 和 VNC

14.1　OpenSSH

传统方式的网络服务或应用程序，如 telnet、ftp、rsh 和 POP 等应用，都是使用明文传输数据，本质上都是不安全的，数据很容易被监听，也很容易受到“中间人”攻击方式的攻击。SSH（Secure SHell）是一项创建在应用层和传输层基础上的安全协议，专为远程登录会话和其他网络服务提供安全性控制，利用 SSH 协议可以有效防止远程处理过程中的信息泄露。目前，SSH 正逐步替代 telnet，并且为 FTP、POP 等服务提供安全通道，但是因为受版权和加密算法等问题的限制，一般都使用开源的 OpenSSH。

14.1.1　OpenSSH 协议

1．OpenSSH 简介

OpenSSH 是 SSH 协议的免费开源实现，是 OpenBSD 的子项目，它首次于 1999 年 10 月出现在 OpenBSD2.6 中，当初的目的是取代由 SSH Communications Security 所提供的 SSH 软件。OpenSSH 可以用在目前几乎所有的操作系统平台上，包括 UNIX、Linux、Windows 等。OpenSSH 已经是目前大多数 Linux 和 BSD 操作系统的标准组件。OpenSSH 支持 SSH 协议的版本 1.3、1.5 和 2。自从 OpenSSH 的 2.9 版本以来，默认的协议是版本 2，协议默认使用 RSA 密钥。OpenSSH 常常被误以为与 OpenSSL 有关系，实际上这两个计划有不同的目的、不同的发展团队，它们的名称相近，只是因为两者有同样的软件发展目标——提供开放源代码的加密通信软件。

2．SSH 的作用

SSH 是一组用于安全访问的远程连接工具，可以替代 rlogin、rsh、rcp 及 telnet 等工具。SSH 对所有的传输进行加密，从而有效地阻止了窃听、连接劫持以及其他网络级的攻击。

SSH 也是一种加密协议，不仅在登录过程中对密码进行加密传送，而且对登录后执行命令的数据也进行加密，这样即使黑客在网络上监听并截获了数据包，也看不懂其中的内容。

14.1.2　软件安装与启动管理

1．软件安装

OpenSSH 现已成为 Fedora 系统的标准组件被默认安装，即使用户在安装操作系统的时候选择“最小安装（Minimal Install）”，所以一般情况下不需要重新安装软件包，只需要更改一些安全设置即可。

在红帽系统中，OpenSSH 的相关软件包有 openssh（SSH 协议版本 1 和 2 的开源实现包）、openssh-clients（客户端）和 openssh-server（服务端）；在 Ubuntu 下只有 openssh-server 和

openssh-client 两个。安装方法如下：

```
#yum install openssh-server openssh openssh-clients                    #红帽系统
#apt install openssh-server openssh-client                             #Ubuntu 系统
```

OpenSSH 包括以下内容。

客户端程序：ssh、scp、sftp 和 slogin
服务端程序：sshd、sftp-server 和 ssh-agent
密钥管理程序：ssh-add、ssh-keysign、ssh-keyscan、ssh-keygen 和 sshd-keygen

用户也可以选择只安装服务器或只安装客户端包。

2．服务管理

Openssh 的服务在红帽和 Ubuntu 下分别为 sshd 和 ssh，可用服务管理工具管理它。
1）红帽系统

```
#systemctl status sshd                                  #检查状态
#systemctl enable/diable sshd                           #启用/禁用/检查状态
#systemctl start/stop/restart/reload sshd               #启动/停止/重启/重载
```

2）Ubuntu 系统

```
#systemctl status ssh                                   #检查状态
#systemctl enable/diable ssh                            #启用/禁用/检查状态
#systemctl start/stop/restart/reload ssh                #启动/停止/重启/重载
```

每当修改配置文件后，都要重启或重载服务，建议重载。

14.1.3　配置文件与服务器设置

1．OpenSSH 主配置文件及默认配置

OpenSSH 的主配置文件为/etc/ssh/sshd_config，可通过命令

```
#grep -v -E "^$|^#" /etc/ssh/sshd_config
```

得到配置文件中的默认配置。经过重新排序的默认配置见表 14-1。

表 14-1　重新排序的默认配置

Fedora	Ubuntu
AcceptEnv LANG LC_CTYPE LC_NUMERIC …	AcceptEnv LANG LC_*
AcceptEnv XMODIFIERS	ChallengeResponseAuthentication no
AuthorizedKeysFile .ssh/authorized_keys	HostbasedAuthentication no
ChallengeResponseAuthentication no	HostKey /etc/ssh/ssh_host_dsa_key
GSSAPIAuthentication yes	HostKey /etc/ssh/ssh_host_ecdsa_key
GSSAPICleanupCredentials no	HostKey /etc/ssh/ssh_host_ed25519_key
HostKey /etc/ssh/ssh_host_ecdsa_key	HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ed25519_key	IgnoreRhosts yes
HostKey /etc/ssh/ssh_host_rsa_key	KeyRegenerationInterval 3600
PasswordAuthentication yes	LoginGraceTime 120

（续表）

Fedora	Ubuntu
PermitRootLogin yes Subsystem sftp /usr/libexec/openssh/sftp-server SyslogFacility AUTHPRIV UsePAM yes X11Forwarding yes	PermitEmptyPasswords no PrintLastLog yes PrintMotd no Protocol 2 PubkeyAuthentication yes RhostsRSAAuthentication no RSAAuthentication yes ServerKeyBits 1024 StrictModes yes Subsystem sftp /usr/lib/openssh/sftp-server TCPKeepAlive yes UsePAM yes UsePrivilegeSeparation yes X11DisplayOffset 10 X11Forwarding yes

通过命令

```
#sshd -T
```

可以输出 sshd 正在使用的所有配置，还可通过 man sshd_config 得到更详尽的配置信息描述。

2．OpenSSH 的默认配置

OpenSSH 安装完毕后，服务程序会自动载入一些配置文件里没有定义的默认配置来确保服务的正常运行，常见默认配置如下。

（1）OpenSSH 默认监听服务器配置所有地址（含 IPv4 地址和 IPv6 地址）上的 TCP 22 端口，用户根据安全需求对这些参数进行修改。

```
Port 22                   #监听端口
AddressFamily any         #制定使用何类地址，有效值有 any，inet(仅 IPv4)或 inet6(仅 IPv6)
ListenAddress 0.0.0.0:22  #监听 IPv4 地址
ListenAddress [::]:22     #监听 IPv6 地址
```

（2）记录有关认证信息。

```
SyslogFacility AUTHPRIV   #记录消息时使用的设施代码
LogLevel INFO             #记录消息时使用的详细程度级别
```

（3）PermitRootLogin 定义了 root 登录的方式，默认设置是允许 root 进行远程登录的，从安全方面考虑，建议将此项设置为 no。

```
PermitRootLogin yes       #可取值有 yes、no、without-password 和 forced-commands-only
```

3．密钥管理

客户机要连接使用 SSH 协议的服务器，除了要有指定用户外，还需要针对此用户的一对密钥：一个公钥，一个私钥。SSH 协议通常采用 RSA 和 DSA 认证算法。ssh-keygen 命令用于密钥管理。

1）RSA 密钥管理

（1）RSA 密钥对的生成。

```
#ssh-keygen -t rsa                         #生成 RSA 密钥对，交互过程如下：
    Generating public/private rsa key pair.
    Enter file in which to save the key (/root/.ssh/id_rsa):
    Enter passphrase (empty for no passphrase):
    Enter same passphrase again:
    Your identification has been saved in /root/.ssh/id_rsa.
    Your public key has been saved in /root/.ssh/id_rsa.pub.
    The key fingerprint is:
    SHA256:1klKRRsNzg04/0rQrtgQyEUIL5+MMn1ckfhKusrRsUM root@ub16clt1. UNIX.gjshao
    The key's randomart image is:
    +---[RSA 2048]----+
    |   .. +o. o*o        |
    |.o                       |
    +----[SHA256]-----+
#ls -l ~/.ssh/
-rw-------. 1 root root 1679 Jun 11 14:06 id_rsa
-rw-r--r--. 1 root root   392 Jun 11 14:06 id_rsa.pub
-rw-r--r--. 1 root root   171 Jun 11 13:29 known_hosts
```

（2）私钥的安装。

私钥存放在用户根目录下的$HOME/.ssh/id_rsa 中。

（3）公钥的安装。

公钥存放在$HOME/.ssh/id_rsa.pub 中。用户应该把 id_rsa.pub 复制到远程服务器中，并将其内容添加到$HOME/.ssh/authorized_keys 文件中。假设 id_rsa.pub 已经复制到位，可在远程系统上执行以下命令来完成：

```
#cat id_rsa.pub   >> ~/.ssh/authorized_keys
```

2）DSA 密钥管理

（1）DSA 密钥对的生成。

```
#ssh-keygen -t dsa                #生成 DSA 密钥对，过程同上，做略
#ls -l ~/.ssh/
```

（2）私钥的安装。

私钥存放在用户根目录下的$HOME/.ssh/id_dsa 中。

（3）公钥的安装。

DSA 公用密钥的安装几乎和 RSA 安装完全一样。公钥存放在$HOME/.ssh/id_dsa.pub 中。用户应该把 id_dsa.pub 复制到远程服务器中，并将其内容添加到文件$HOME/.ssh/authorized_keys 之中。

4．启动 sshd

在设置好相应的配置参数后，启动 sshd 服务，方法如下：

1）红帽系统

```
#systemctl enable sshd          #启用 ssh 服务
#systemctl restart sshd         #启动 ssh 服务
```

2）Ubuntu 系统

```
#systemctl enable ssh           #启用 ssh 服务
#systemctl restart ssh          #启动 ssh 服务
```

若已经启用或启动则不需重做。

14.1.4　客户端及应用

1．远程登录命令（ssh/slogin）

ssh/slogin 是远程登录客户端程序，允许用户在远程机器上登录并在其上执行命令。ssh/slogin 的用法为：

```
#ssh/slogin [options] [-l user] hostname/IP [cmd]
#ssh/slogin [options] user@hostname/IP [cmd]
```

ssh/slogin 的部分常用参数见表 14-2。

表 14-2　ssh 的部分常用参数

参　数	功 能 描 述	参　数	功 能 描 述
[-l] user	指定用户名，若不指定则使用当前用户	hostname/IP	ssh 服务器的主机名或 IP
cmd	远程主机上的 shell 命令	-p port	指定端口号
-q	安静模式，抑制不必要的输出	-v	显示调试信息
-F file	指定用户配置文件为 file，忽略默认配置	-I identity_file	指定本地私钥文件,忽略默认
-1	使用版本 1	-2	使用版本 2
-4	使用 IPv4	-6	使用 IPv6

第一次使用 ssh 于远程服务器上登录时，会出现如下提示信息：

```
The authenticity of host '********' can't be established.
*** key fingerprint is SHA256:iplI8Dkd/GevN+uuSY1n+FB67/EqXOzrWHGOObk4kyM.
*** key fingerprint is MD5:96:3d:83:20:2e:7c:15:c9:ae:ab:a3:06:43:9f:b1:72.
Are you sure you want to continue connecting (yes/no)?
```

键入 yes 继续，会将该服务器添加到已知主机的列表中，如下面的消息所示：

```
Warning: Permanently added '********' (RSA) to the list of known hosts.
```

下一步，将会看到询问远程主机口令的提示：

```
…'s password:
```

在输入正确口令后，用户会登录到远程主机，并出现 shell 提示符。如果没有指定用户名，

则使用与本地用户同名的用户。如果想指定不同的用户名，使用下面的命令方式：

```
#ssh user@hostname/hostIP                    或
#ssh [-l] user hostname/hostIP
```

ssh 命令可以用来在远程机器上不经登录而执行命令，其一般用法为：

```
#ssh user@hostname/hostIP cmd
```

当然，认证是少不了的。比如，如果想以长格式列出远程主机 example.host 当前用户家目录的内容，可以输入以下命令：

```
#ssh example.host ls -l
```

如果要显示指定用户 test 家目录的内容，可以输入以下命令：

```
#ssh test@example.host ls -l
```

2．远程文件复制（scp）

scp 命令可以通过安全、加密的链接在机器间复制文件，意为安全地复制。scp 在需要时会提示输入对方密码或密钥，也允许在两个异地主机间复制文件。scp 的一般语法为：

```
scp sources destination
```

source 用于指定源文件，destination 用于指定目的文件。当其中之一为本地文件时，可以直接使用本地方式的文件，可以是相对路径的，也可以是绝对路径的。当文件为异地文件时所使用的文件名格式为：

```
username@hostname:filename         或
username@hostIP:filename
```

当然，scp 还有一些用于传输控制的选项，用户可以参阅 scp 的 man 手册。

1）从本地向异地复制

若把本地文件 myproc.c 复制到 example.host 上的账号为 test 的用户家目录内，可以在 shell 提示符下输入：

```
#scp    myproc.c    test@example.host:myproc.c
```

将本地当前目录下的所有 C 程序*.c 复制到 example.host 上 test 的用户家目录内，可以在 shell 提示符下输入：

```
scp *.c    test@example.host:.               #最后的“.”不能少
```

2）从异地复制到本地

将 example.host 主机上用户为 test 的文件 Download/music.mp3 复制到本地系统的临时目录，可以使用命令：

```
#scp test@example.host:Download/music.mp3 /tmp/music.mp3
```

3）异地到异地复制

将异地主机 example.host 上 test 用户的文件 Download/music.mp3 复制到异地主机 example2.host 用户 test2 的家目录内，并更名为 music1.mp3，可使用命令：

#scp test@example.host:Download/music.mp3 test2@example2.host:music1.mp3

3．安全的 ftp（sftp）

像 FTP 一样，sftp 是一个交互式文件传输命令，允许使用标准的 ftp 命令进行传输与控制，它的传输是基于 ssh 加密机制的，并可使用 ssh 的公钥认证等特征。sftp 的常用方法如下：

```
sftp user@hostname.com            或
sftp user@hostIP
```

一旦通过验证，可以使用一组和 ftp 相似的命令，详细用法及内部命令参阅 sftp 的 man 手册。

14.1.5　与防火墙和 SELinux 的关系

1．与防火墙的关系

为了能更好地使用远程登录服务，在防火墙规则的设置上，要允许 ssh/22 通过。设置方法如下：

```
#firewall-cmd --add-service ssh                #firewalld 防火墙
#ufw allow ssh                                 #ufw 防火墙
```

2．与 SELinux 的关系

1）进程类型

SELinux 为系统上运行的每个进程定义进程类型，服务进程 sshd 的进程类型为 sshd_t，可以使用 ps 命令中的-Z 选项来查看进程的上下文。例如：

```
#ps -eZ | grep sshd                 #输出如下
system_u:system_r:sshd_t:s0-s0:c0.c1023 1007 ? 00:00:00 sshd
system_u:system_r:sshd_t:s0-s0:c0.c1023 2655 ? 00:00:00 sshd
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 2669 ? 00:00:00 sshd
```

可使用命令：

```
#semanage permissive -a sshd_t
```

使类型为 sshd_t 的进程不会被 SELinux 拒绝访问。

2）布尔变量

SELinux 对 SSH 所使用的控制变量有：

ssh_keysign：是否允许基于主机密钥的身份验证
ssh_chroot_rw_homedirs：是否允许具有 chroot env 的 ssh 在用户主目录中读取和写入文件
ssh_sysadm_login：是否允许 ssh 作为 sysadm_r:sysadm_t 登录
sftpd_write_ssh_home：是否允许 sftpd 写家目录

可以通过 setsebool 命令来更改这些变量的值，例如：

```
#setsebool -P ssh_keysign 1
```

14.2　VNC

VNC（Virtual Network Computing）是一种使用 RFB 协议的屏幕画面分享及远端操作软件，此软件可通过网络发送键盘与鼠标的动作和即时的屏幕画面。VNC 系统由客户端（也称为 Viewer）、服务端和一个协议组成，服务端被动接受客户端的控制，分享其所运行计算机的屏幕，客户端与服务端交互，观察控制服务端；VNC 服务负责将服务端原始图像传送到客户端并将客户端的事件消息传送到服务端。

VNC 可以跨平台使用，支持 UNIX、Linux、Windows 等操作系统，甚至在没有安装客户端程序的电脑中，只要有支持 Java 的浏览器也可使用。2002 年，VNC 以 GPL 发布，因其免费、简单、可靠、向后兼容性及操作系统无关性等特点，使之成为最为广泛使用的远程控制软件，使得网络管理员可以使用一种工具来管理几乎所有的操作系统。

VNC 软件有多种版本，比如，红帽系统有 TigerVNC，Ubuntu 有 x11vnc 和 vnc4server 等，这里我们只简单介绍 TigerVNC 和 X11VNC 的用法。

14.2.1　红帽的 TigerVNC

1．安装前准备

安装 VNC 服务端前，服务器需要有一个可用的桌面环境，如果没有，需先安装一个。

```
#yum check-update                              #检查系统中所有软件包的更新
#yum -y groupinstall "MATE Desktop"            #比如，安装 MATE 桌面
```

安装完毕后，启动图形界面：

```
#echo "exec /usr/bin/mate-session" >> ~/.xinitrc
#systemctl set-default graphicl.target         #设置系统启动后接入图形界面
#startx                                        #启动图形界面
```

不过，我们介绍的系统都是基于桌面或工作站的，不需要再安装桌面系统。

2．TigerVNC 服务包的安装

TigerVNC 服务包含 tigervnc-license、tigervnc-server-minimal 和 tigervnc-server 三个安装包，在一般情况下，系统已经安装 tigervnc-license、tigervnc-server-minimal 两个包。安装 tigervnc-server 的方法如下：

```
#yum -y install tigervnc-server
```

3．VNC 服务的启动和关闭

（1）启动 VNC 服务。

```
#vncpasswd          #修改当前用户 VNC 登录密码，生成~/.vnc/passwd
#vncserver   :1     #启动 VNC 服务，:1 表示对应桌面号；生成~/.vnc/xstartup 和~/.vnc/config
```

VNC 服务第一次启动时，会判断当前用户是否设置了独立 VNC 登录密码，若没有，则会要求用户设置。同时会在~/.vnc/目录中生成 xstartup 和 config 文件。启动之后，就可以使用 vnc 客户端登录了。

VNC 服务使用的端口号与桌面号相关，VNC 使用 TCP 端口从 5900 开始，服务为 rfb。

（2）关闭 VNC 服务。

```
#vncserver -kill :1
```

关闭之后，可以修改 xstartup 文件，使之能更好地满足要求。对于刚安装的 MATE 桌面，可将最后一行 exec /etc/X11/xinit/xinitrc 修改为.xinitrc 相同的配置，具体可为：

```
exec /usr/bin/mate-session
```

对于 GNOME 桌面，可将最后一行修改为：

```
exec /usr/bin/gnome-session                                 #GNOME3    或
exec /usr/bin/gnome-session --session gnome-classic         #经典界面
```

4．VNC 服务的配置

手动启动 VNC 服务虽然简单，但每次系统重启后，需要再次手动启动 VNC 服务，给管理员带来不必要的麻烦。配置 tigervnc 服务自动启动的方法如下：

（1）创建 vncserver 服务 unit。

```
#cp /lib/systemd/system/vncserver@.service /etc/systemd/system/vncserver@:1.service
```

（2）修改配置文件，并管理 vncserver 服务。

```
#vi /etc/systemd/system/vncserver@:1.service    #将[Service]部分的内容替换如下
[Service]
Type=forking
User=root
ExecStart=/usr/bin/vncserver :1 -geometry 1280x1024 -depth 24
PIDFile=/root/.vnc/%H:1.pid
ExecStop=/usr/bin/vncserver -kill :1
```

保存完毕后，运行如下命令：

```
#systemctl daemon-reload                #重载 systemd 配置
#systemctl enable vncserver@:1          #设置 vnc 服务自启
#systemctl start vncserver@:1           #启动 vnc 服务
#systemctl status vncserver@:1          #查看 vnc 服务运行状态
```

5．防火墙设置

虽然 VNC 服务启动了，但外部用户仍然无法访问，需要配置防火墙让其允许外部用户访问 VNC 服务。

```
#firewall-cmd --add-service=vnc-server --permanent          #添加到防火墙规则
#firewall-cmd --reload
```

14.2.2　客户端及使用

VNC 的客户端是通用的，有的系统提供多个，有的系统没有提供，这里我们使用 https://www.realvnc.com/download/viewer 上的 realvnc，有 Windows、Linux、MACOS 和 UNIX 等多种平台，下载后可作为学习和实验之用。

realvnc 在 Windows 系统的程序名为 VNC-Viewer-6.1.1-Windows-*bit.exe，在 Linux 系统下的程序名为 VNC-Viewer-6.1.1-Linux-x*，必要时为其增加执行权。执行方法是：

```
#chmod +x VNC-Viewer-6.1.1-Linux-*          #添加执行权
$ ./VNC-Viewer-6.1.1-Linux-*                #需要在地址栏输入地址:编号
$ ./VNC-Viewer-6.1.1-Linux-*  地址:编号      #比如
$ ./VNC-Viewer-6.1.1-Linux-* 192.168.137.24:1
```

如图 14-1 所示为登录 Fedora 24 时通过密码验证后传统的 GNOME 界面。

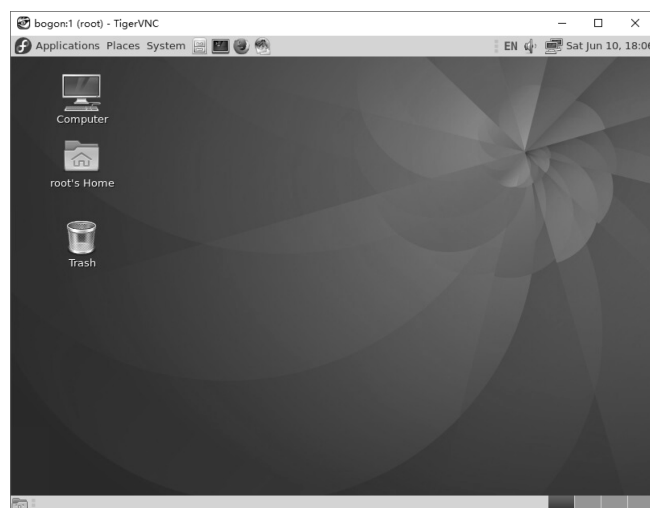


图 14-1　登录 Fedora 24 成功后传统 GNOME 界面

14.2.3　Ubuntu 的 x11vnc

1．x11vnc 服务包的安装

Ubuntu 的 x11vnc 服务软件包为 x11vnc，安装方法如下：

```
#apt install x11vnc -y
```

2．服务的启动和关闭

1）生成密码

```
#x11vnc -storepasswd          #修改当前用户 VNC 登录密码，生成~/.vnc/passwd
```

2）启动服务

```
#/usr/bin/x11vnc -ncache 10 -auth guess -forever -loop -noxdamage -repeat -rfbauth /etc/x11vnc.pass
-rfbport 5900 -shared
```

3）创建服务 unit 文件

编辑文件/lib/systemd/system/x11vnc.service，在其中添加如下内容：

```
[Unit]
Description=Start x11vnc at startup.
After=multi-user.target
[Service]
Type=simple
ExecStart=/usr/bin/x11vnc -ncache 10 -auth guess -once -loop -noxdamage -repeat -rfbauth
/root/.vnc/passwd -rfbport 5900 -shared
[Install]
WantedBy=multi-user.target
```

4）重载 systemctl 配置

```
#systemctl daemon-reload
```

5）启用和启动 x11vnc

```
#systemctl enable x11vnc.service
#systemctl start x11vnc.service
```

3．防火墙设置

```
#firewall-cmd --add-port=5900/tcp --permanent          #firewalld 防火墙
#ufw allow port 5900                                   #ufw 防火墙
```

习题 14

1．简述题

（1）简述 ssh 的功能及与传统 telnet 等的区别。
（2）简述 OpenSSH 主配置文件的选项及含义。
（3）简述 VNC 的作用。

2．综述题

综述 ssh 在网络中的应用。

实验 14

1．使用 ssh 进行密钥管理并进行远程登录，使用 scp、sftp 进行远程文件安全复制。
2．在 Linux 下配置 VNC 服务器，并跨系统使用 VNC 客户端。

第 15 章　Web 服务器 Apache

15.1　Apache 概述

Apache 是世界使用排名第一的 Web 服务器软件。据著名的 Web 服务器调查公司 Netcraft 2017 年 1 月发布的调查报告显示，活跃站点市场份额占 46.3%。世界上很多著名的网站，如 Yahoo、Amazon.com、W3 Consortium、Financial Times 等都是 Apache 的产物。它的成功之处主要在于它开放源代码、拥有一支开放的开发队伍、支持跨平台的应用以及它的可移植性等。它快速、可靠并且可通过简单的 API 扩展，将 Perl/Python 等解释器编译到其中。它可以运行在几乎所有广泛使用的计算机平台上，如 UNIX、Windows、Linux 系统等。

Apache 源于 NCS 的 Ahttpd 服务器，取自“a patchy server”的读音，意思是充满补丁的服务器，它是自由软件，所以不断有人自发为它开发新的功能、特性，修改原来的缺陷。Apache 以其丰富的功能和优越的性能，加上与 Linux 的完美结合，备受广大用户的喜爱。HP、SUN 等大公司分别在自己的操作系统中捆绑了 Apache 作为 Web 服务器软件。

Apache 采用了模块化的体系结构，不仅提供了基本的 Web 服务，而且还具有极强的可扩展性。它拥有以下特性：

- 开源软件，可免费获取；
- 支持最新的HTTP通信协议；
- 可运行于几乎所有的计算机操作系统上；
- 配置文件简单，容易操作；
- 支持实时监视服务器状态和定制服务器日志；
- 支持 IPv6；
- 支持多种方式的 HTTP 认证；
- 支持 Web 目录修改；
- 支持 Perl、PHP 等 CGI 脚本；
- 支持服务器端，包含指令（SSI）；
- 支持安全 Socket 层（SSL）；
- 支持 FastCGI；
- 支持基于 IP、主机名和端口号三种类型的虚拟主机服务；
- 跟踪用户会话；
- 支持动态共享对象；
- 支持多进程和多进程混合模型 MPM；
- 支持第三方软件开发商提供的功能模块。

15.2　Apache 的安装和启动

15.2.1　Apache 的安装

在红帽系统中，与 Apache 相关的有以下几个软件包：httpd（服务器软件包）、

httpd-filesystem（基本目录布局包）、httpd-tools（服务器工具软件包）和 httpd-manual（用户帮助手册），其中的 httpd-manual 是可选的，可以不安装，其他相关的包会在安装时自动被安装。

用户可以使用以下命令来判断 Apache 是否已经安装：

```
#rpm -qa | grep httpd                    #判断 Apache 是否安装
```

或

```
#yum list installed httpd*
```

也可以使用 dnf 命令来判断：

```
#yum list httpd                          #列出所有仓库中所有可用的软件包和已安装的软件包
```

若还没有安装，可以使用软件包管理的办法安装它们。假设已经联网，可用 yum 或 dnf 命令安装它们，方法是：

```
#yum install httpd httpd-manual          #安装 httpd 及相关包。Fedora 上可以使用 dnf
```

Ubuntu 系统的 apache 服务器软件包为 apache2，还有一个文件档包 apache2-doc，可采用 ubuntu 软件包管理办法安装它。可首先检查 apache2 是否已经安装。

```
#dpkg -l apache2*                        ##判断 Apache 是否安装，或
#apt list --installed apache2*
```

假设已经联网，可用 apt 或 apt-get 命令安装它们，方法是：

```
#apt installed apache2 apache2-doc       ##安装 apache2 及其文档。
```

15.2.2　Apache 服务器启动管理

红帽与 Ubuntu 系统 Apache 服务器的服务名分别为 httpd.service 和 apache.service，可用服务启动管理的命令 systemctl 进行管理。

1．运行状态检查

```
#systemctl status httpd                  #红帽系统
#systemctl status apache2                #Ubuntu
#apachectl status                        #所有系统
```

2．设置开机时自启动

```
#systemctl enable httpd                  #红帽系统
#systemctl enable apache2                #Ubuntu
```

3．立即启动

```
#systemctl start| stop | restart httpd   #红帽系统
#systemctl start| stop | restart apache2 #Ubuntu
#apachectl start | stop | restart        #所有系统
```

4．检查配置文件语法是否正确

```
#httpd -t                                #红帽系统
```

```
#apache2 -t                    #Ubuntu
#apachectl configtest          #所有系统
```

5．查看已经被编译到 Apache 中的模块

```
#httpd -1                      #红帽系统
#apache2 -l                    #Ubuntu
```

15.2.3　Apache 服务的测试

在默认情况下，网站的文档目录为/var/www/html，默认文档为 index.html，为了避免冲突，我们在/var/www/html 中创建一个文档名为 index2.html，内容如下：

```
<HTML>
  <TITLE>Apache Server Test!</TITLE>
  <BODY> <H2>Hello Apache World!</H2> </BODY>
</HTML>
```

使用文本浏览器 elinks 来测试，在屏幕上显示“Hello Apache World!”，表明 Apache 服务器正常配置并已经启动。

```
#elinks http://127.0.0.1/index2.html      #若成功，输入 q 并回车退出。需要时，先安装 elinks 包
```

或者使用 IE 等浏览器，在地址栏中输入 http://127.0.0.1/index2.html。

15.3　Apache 的配置文件和配置指令

在红帽系统中，Apache 服务的默认配置目录为/etc/httpd/conf/，主配置文件为 httpd.conf。在 Ubuntu 系统中，Apache 服务的默认配置目录为/etc/apache2，主配置文件为 apache2.conf。尽管名字不同，所处位置也不同，但配置方法是一样的。在一般情况下，通过对主配置文件的修改来实现 Apache 的各种功能。

主配置文件 httpd.conf 或 apache2.conf 可被拆分成多个文件，然后通过 Include 和 IncludeOptional 将它们整合到主配置文件中来。

15.3.1　Apache 主配置文件的结构

在 Fedora 中，Apache 默认配置文件 300 余行，但是其中只有不到 30%真正起作用，很多的配置行是作为注释形式出现的。需要注意的是，这些注释有很好的帮助，而且有很多行只要去掉前头的注释标记“#”后直接可以起作用，是否让其起作用，要视具体需要而定。

Apache 的配置文件分三大节，它们分别是：

（1）全局配置节（Global Environment）。用于控制 Apache 服务器进程的全局操作。

（2）主服务器或默认配置节（Main or Default Server）。用于处理任何不被<VirtualHost>段处理的请求，也就是服务器默认配置，这里的参数都可以写在虚拟主机段中。

（3）虚拟机配置节（Virtual Hosts）。用于为用户自设的虚拟主机提供配置。

用户可根据需要，修改配置文件以建立符合自己需要的 Apache 服务器。

15.3.2　Apache 的配置指令

现以默认主配置文件为例，详细说明以下各配置指令。

1．全局配置节指令

（1）ServerTokens OS：服务器回应给客户端的"Server:"应答头，包含关于服务器操作系统类型和编译的模块描述信息。

（2）ServerRoot dir：设置服务器配置目录（需要注意的是路径不能以“/”结尾）。在红帽系统中为"/etc/httpd"，在 Ubuntu 下为"/etc/apache2"。

（3）PidFile file：指定 Apache 服务器的进程号文件号文件及存放的位置，此处文件是相对于 ServerRoot 指定的值取放的。在红帽系统为 run，被链接到/run/httpd；Ubuntu 使用默认。

（4）Timeout scnds：指定发送或接收超时间隔为 scnds 秒，比如 120。

（5）KeepAlive Off/On：设置是否允许保持连接，若值为 Off，表示不允许保持连接，此时一次连接只能响应一个请求，而一个请求一般情况下只传输一个文件，这种配置能节省大量内存，但增加了系统处理 TCP 握手和关闭的操作，会多消耗一些 CPU 资源。若设为 On，则表示允许保持连接，此时一次连接可以连续响应多个请求，这在一个页面上有很多相关文件的情况下，可以明显提高响应速度，但同时也意味每次用户完成全部访问后，都要保持一定时间后才关闭 TCP 连接，那么在关闭连接之前，必然会有一个 Apache 进程对应该用户而不能处理其他用户，在这种配置中，系统少处理了很多 TCP 的握手和关闭的操作，减少了 CPU 资源的消耗，但却会多消耗服务器的内存。

（6）MaxKeepAliveRequests N：设置一次保持连接最多包含的请求数，比如 100，若不想限制就设置为 0。

（7）KeepAliveTimeout scnds：设置一次保持连接的超时间隔秒数，比如 15。

（8）服务器池设置。在 Apache 2.0 中提供了三种服务器的工作方式。

prefork MPM：一个父进程，然后根据设置及连接情况生成相应的子进程数。这种模式可靠性和健壮性都是最好的。但是在性能上，开销过大。管理员应选择 prefork MPM 来实施基于流程的 Web 服务器。虽然基于流程的服务器有一些速度较慢的性能，它们通过模块来提供稳定性和兼容性，这些模块不支持线程。Linux 环境下多采用这种模式。

worker MPM：混合线程/进程的 MPM。一个父进程，后面是带有线程的子进程，每个子进程的线程数是固定且相同的。它提供比 prefork MPM 更卓越的可靠性、故障恢复能力和可扩展性。大多数 Apache 1.3 模块不是线程安全，并且由于 worker MPM 不向后兼容，因此，使用这一 MPM 的管理员也不能一同使用 Apache 1.3 模块和 Apache 2.0。该模式主要用于 UNIX 环境。

perchild MPM：也是混合线程/进程的 MPM。当启动 perchild MPM 时，它会建立指定数量的子进程，且每个子进程都具有指定数量的线程，如负载增加了，那它不会建立新的进程（子进程是固定的），只是在子进程下建立新的线程。它还有一个特点就是可以为每一个子进程配置不同的用户和组，也可以为每个虚拟主机指定一个子进程。

以下是对 prefork MPM 和 worker MPM 的描述。

```
#prefork MPM
<IfModule prefork.c>
    StartServers 8              //设置 Apache 服务器运行时，启动的服务器子进程个数
    MinSpareServers 5           //设置最小空闲服务器子进程的个数
    MaxSpareServers 20          //设置最大空闲服务器子进程的个数
    ServerLimit 256             //设置服务器生命周期内 MaxClients 的最大值
    MaxClients 256              //设置同时响应的客户端数量
    MaxRequestPerChild 4000  //设置每个服务器子进程最多可以服务的请求数。到达
MaxRequestsPerChild 的限制后，子进程将会结束。为 " 0 " 时，子进程将永远不会结束
#worker MPM
</IfModule>
<IfModule worker.c>
    StartServers 2              //设置服务器初始子进程的个数
    MaxClients 150
    MinSpareThreads 25
    MaxSpareThreads 75
    ThreadsPerChild 25          //设置每个服务器进程的工作线程的数量
    MaxRequestsPerChild 0
</IfModule>
```

（9）Listen port：设置 Apache 服务器监听端口。比如

```
Listen 80                       #80 为默认端口
```

（10）动态共享对象支持

```
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authn_alias_module modules/mod_authn_alias.so
…
LoadModule cgi_module modules/mod_cgi.so
```

Apache 采用模块化的结构，各种可扩展的特定功能以模块形式存在而没有静态编入 Apache 的内核，这些模块可以动态载入 Apache 服务过程中。这样在很大程度上方便了 Apache 功能的丰富和完善。

（11）Include file：将 file 包含到主配置文件中。此处文件是相对于 ServerRoot 存在的。这是 Apache 配置文件的灵活之处，使得不需要修改主配置文件就能对服务器进行配置。比如 conf.d/*.conf 将 conf.d 目录下所有以.conf 结尾的文件都引入配置中。

（12）ExtendedStatus On/Off：Apache 服务器可以通过特殊的 HTTP 请求来报告自身的运行状态，在默认情况下是 Off。该指令需要后面的参数段<Location /server-status>相配合。比如 ExtendedStatus On 可以让服务器报告更全面的运行状态信息。

（13）Apache 服务器子进程的运行身份及属组。

```
User apache：指定 Apache 服务器子进程运行时的身份为 apache。
Group apache：指定 Apache 服务器子进程运行时的属组为 apache。
```

2．主服务器或默认配置节指令

（1）ServerAdmin root@localhost：指定 Apache 服务器管理员的邮箱。

（2）ServerName www.example.com:80：设置 Apache 服务器默认站点的名称和端口号。www.example.com 可以是 IP 地址，也可以是域名。若指定的是域名，需要服务器所有网络中有 DNS 服务器作解析。

（3）UserCanonicalName On/Off：设置是否使用规范名称。若为 Off，表示使用由客户提供的主机名和端口号；若为 On，表示使用 ServerName 指令设置的值。例如，如果有一个 Web 服务器，其中 ServerName 设置的值是 www.hncj.edu.cn，当用户输入 http://www/index.html 时，有以下两种情况：

UserCanonicalName 值为 Off 时，相当于 http://www/index.html；
UserCanonicalName 值为 On 时，相当于 http://www.hncj.edu.cn/index.html。

（4）DocumentRoot “/var/www/html”：设置默认站点的文档根目录。

（5）根目录的访问控制。

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
```

在“主服务器或默认配置节”中有很多个 Directory 参数配置段，它的写法类似 HTML 的格式，<Directory/>表示开始，</Directory>表示结束，其间的内容要对文件系统的目录进行限制。

Options FollowSymLinks 表示允许跟随符号链接，具体 Options 的值参见表 15-1。

AllowOverride None 表示不允许覆盖，AllowOverride 的值参见表 15-2。

表 15-1　Options 选项值及含义

选　　项	功　　能
All	默认值，除最后 3 个值外允许所有操作
ExecCGI	允许执行 CGI
FollowSymLinks	允许跟符号链接
Includes	允许服务端包含 SSI
Indexes	允许显示文件目录清单
IncludesNOEXEC	允许 SSI，但取消 CGI 的#exec 和#include
Multiviews	允许对资源的不同表现形式进行协商，如语言
SymLinksIfOwnerMatch	允许符号链接到其他文件目录，但是所有都必须匹配

表 15-2　AllowOverride 配置项及其含义

控　制　项	典型可用指令	功　　能
None	禁止使用全部指令	禁止处理.htaccess 文件
All	允许全部指令	允许全部指令

（续表）

控　制　项	典型可用指令	功　　能
AuthConfig	AuthName,AuthType,AuthUserFile,Require	进行认证、授权的指令
FileInfo	DefaultType,ErrorDocument,sethander	控制文件处理方式的指令
Indexes	AddIcon,DefaultIcon,HeaderName DirectoryIndexes	控制目录处理方式的指令
Limit	Allow,Deny,Order	进行目录访问控制的指令
Options	Options,XBitHack	启用不能在主配置文件中使用的各种选项

（6）文档根目录的访问控制。

<Directory “/var/www/html”>

Options Indexes FollowSymLinks //设置允许与随同符号链接。Indexes 的含义是如果要访问的文档不存在，则会显示该目录下的文件目录清单

AllowOverride None　//不允许覆盖当前配置，即不处理.htaccess 文件

Order allow,deny　　//设置访问控制的顺序，本句表示先执行 allow 指令，再执行 deny 指令。使用该选项配合后面的 Allow 和 Deny 指令，可以实现灵活的访问控制

Allow from all　　　//允许从任何地点访问目录

</Directory>

（7）开放个人主页。

<IfModule mod_userdir.c>

UserDir disabled　　//允许/禁用个人主页功能，值为 enabled 时表示允许，值为 disabled 时表示禁用，值后面还可以跟具体的用户，表示允许/禁用某个用户的主页。

UserDir public_html　//指明个人主页的文档目录名称为 public_html。

</ IfModule >

假设本地服务器的域名为 www.hncj.edu.cn，如果 UserDir 处于 enabled 状态，当访问 http://www. hncj.edu.cn/user01/时，相当于访问 www.hncj.edu.cn 站点中的文件/home/user01/public_html/ index.html，即 user01 的个人主页。

除了上述用法外，还有其他用法，下面是两个配置示例。

示例 1：只禁用 user01 的个人主页功能，开放其他用户的个人主页功能。

```
<IfModule mod_userdir.c>
    UserDir disabled user01
    UserDir public_html
</ IfModule >
```

示例 2：只开放 user01 的个人主页功能，禁用其他用户的个人主页功能。

```
<IfModule mod_userdir.c>
    UserDir disabled
    UserDir enabled user01
    UserDir public_html
</ IfModule >
```

（8）对每个用户的个人主页的根目录进行限制

```
<Directory /home/*/public_html>          //此处用了通配符“*”来配置任何用户，表示对每个用户
                                         //的个人主页的根目录进行限制。
    AllowOverride FileInfo AuthConfig Limit    //允许覆盖 FileInfo、AuthConfig、Limit 这 3 项的配置

    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec       //设置该目录具有
                       //MultiViews、Indexes、SymLinksIfOwnerMatch、IncludesNoExec 属性
    <Limit GET POST OPTIONS>            //Limit 指令仅针对 http 的方法进行限制，在这里是限
                                        //制 http 中的 GET、POST 和 OPTIONS 方法
      Order allow,deny
      Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>   //除了 LimitExcept 指令中列出的方法外，对其他未列
                                     //出的方法进行限制
      Order deny,allow
      Deny from all
    </LimitExcept>
</Directory>
```

在一般情况下，当开放个人主页功能时同时启用该配置段。默认此部分是被注释掉的，去掉各行前的“#”即可。

（9）DirectoryIndex index.html index.html.var：指定每个目录的默认文档名。index.html.var 是内容协商文档，在一般情况下，它挑选一个与客户端的请求最符合的文档来响应客户，如语言一致。

（10）AccessFileName .htaccess：指定每个目录中访问控制文件的名称为.htaccess。

（11）文件访问控制

```
<Files ~  " ^\.ht " >
    Order allow,deny
    Deny from all
</Files>
```

作用是针对以.ht 开关的文件进行访问控制，这里表示禁止访问以.ht 开头的文件。

（12）TypesConfig /etc/mime.types：指定 mime.types 文件的存放位置。mime.types 文件中存放了 mime 定义的各种文件类型。

（13）DefaultType text/plain：指定默认的文件类型。

（14）mime_magic 模块

```
<IfModule mod_mime_magic.c>
    MIMEMagicFile conf/magic
</IfModule>
```

作用是通过该模块使得服务器可以按照文件内容中的各种提示信息来决定文件的类型。

（15）HostnameLookups On/Off：用来设置在记录日志时是记录客户机的名称还是 IP 地址，值为 On 时表示记录客户机的名称，为 Off 表示记录客户机的 IP 地址。为了提高速度、

降低流量，也为了便于日志的查询，一般将该值设置为 Off。

（16）EnableMMAP On/Off：指示 httpd 在递送中如果需要读取一个文件的内容时，它是否可以使用内存映射。默认时，如果操作系统支持，Apache 将默认使用内存映射。

（17）EnableSendfile On/Off：控制 httpd 是否可以使用操作系统内核 sendfile 的支持来将文件发送到客户端。在默认情况下，如果操作系统支持，Apache 将使用 sendfile 将文件内容直接发送到客户端而并不读取文件。

（18）ErrorLog logs/error_log：指定错误日志的存放位置。

（19）LogLevel warn：指定日志记录的级别。可能的值包括 debug，info，notice，warn，error，crit，alert，emerg。

（20）日志格式：可以通过以下方式指定日志格式。

```
LogFormat  "%h %l %u %t \"%r\"  %>s %b \"%{Referer}i\"  \"%{User-Agent}i\""  combined
LogFormat  "%h %l %u %t \"%r\"  %>s %b"  common
LogFormat  "%{Referer}i -> %U"  referer
LogFormat  "%{User-agent}i"  agent
```

（21）指定访问日志的位置和类型。

```
CustomLog logs/access_log common：如果没有在虚拟主机内定义这个指令，传输日志将记录在这
里；如果在虚拟主机内使用这个指令，则日志将记录在指定的位置
CustomLog logs/referer_log referer          //如果要单独记录的 referer 日志
CustomLog logs/agent_log agent              //如果要单独记录的 agent 日志
如果要记录单独的 agent 和 referer 日志，可以使用此两条指令，否则注释掉它们。
CustomLog logs/access_log combined          //在一个日志文件中记录 access，agent 和 referer 信息。
```

（22）ServerSignature On/Off：在服务器中产生的页面中增加一行，该行包括服务器的版本和虚拟主机的名称。允许的值有 On、Off、Email。若设为“EMail”将包含一个指向 ServerAdmin 的 mailto:连接。

（23）定义目录别名。

```
Alias /icons/  "/var/www/icons/"       //定义“/var/www/icons/”的别名主“/icons/”。
<Directory  "/var/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

一般情况下，在定义了目录别名后，要对目录别名的访问控制进行配置。

（24）WebDAV 模块配置项。

```
<IfModule mod_dav_fs.c>
    DAVLockDB /var/lib/dav/lockdb
</IfModule>
```

作用是指定 WebDAV 加锁数据库的位置。当 WebDAV 功能激活后，允许客户端通过 Web 方式维护站点的内容。

（25）定义 CGI 脚本目录别名，配置 CGI 模块。

```
ScriptAlias /cgi-bin/  " /var/www/cgi-bin/ "
<Directory  " /var/www/cgi-bin " >
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

定义脚本目录别名，并配置 CGI 模块以及目录的访问限制配置。

（26）设置自动生成目录列表的显示方式。

```
IndexOptions FancyIndexing VersionSort NameWidth=* HTMLTable
```

其中 3 个值的含义如下：

FancyIndexing：在每种类型的文件前加一个小图标以示区别。
VersionSort：对同一个软件的多个版本进行排序。
NameWidth=*：文件名字段自动适应当前目录下的最长文件名。

（27）图标显示。

AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip：用于指定服务器通过 MIME 编码格式来判别文件类型并显示图标。

```
AddIconByType (TXT,/icons/text.gif) text/*
…
AddIconByType (VID,/icons/movie.gif) video/*
```

用于指定服务器通过定义的 MIME 类型来显示图标。

```
AddIcon /icons/binary.gif .bin .exe
…
AddIcon /icons/blank.gif ^^BLANKICON^^
```

用于指定服务器通过文件的扩展名来显示图标。

（28）定义默认图标。

DefaultIcon /icons/unknown.gif：定义默认图标为 unknown.gif

（29）列目录时给出附加内容。

ReadmeName README.html：在进行目录列表时，会将位于该目录下的 README.html 的内容追加到目录列表末尾。

HeaderName HEADER.html：在进行目录列表时，会将位于该目录下的 HEADER.html 的内容追加到目录列表的开头。

（30）定义不参与索引的文件名。

```
IndexIgnore .??* *~ *#HEADER* README* RCS CVS *,v *,t
```

（31）语言支持。

DefaultLanguage nl：定义默认语言
AddLanguage ca .ca　　　　//以下为增加语言支持
AddLanguage cs .cz .cs
…
AddLanguage zh-TW .zh-tw

（32）定义语言优先级。

LanguagePriority en ca cs da de el eo es et fr he hr it ja ko ltz nl nn no pl pt pt-BR ru sv zh-CN zh-TW

（33）指定强制语言优先级的方法。

ForceLanguagePriority Prefer Fallback

含义如下：

Prefer：当有多种语言可以匹配时，使用 LanguagePriority 列表中的第 1 项。
Fallback：当没有语言可以匹配时，使用 LanguagePriority 列表中的第 1 项。

（34）字符集设置。

AddDefaultCharset UTF-8：设置默认字符集为 UTF-8。

（35）增加 MIME 文件类型。

AddEncoding x-compress Z
AddEncoding x-gzip gz tgz

表示允许某些浏览器可以在线解压指定类型的文件。

AddType application/x-tar .tgz
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz

作用与 AddEncoding 相对应，表示定义这些扩展名来指明文件的类型。

（36）定义 AddHandler 映射。

AddHandler type-map var：AddHandler 允许映射确定的文件扩展名到“handlers”，这既能编译到服务器中也可以添加到 Action 指令中。

AddHandler cgi-script .cgi
AddHandler send-as-is asis

（37）关于 SSI 的设置。

AddType text/html .shtml
AddOutputFilter INCLUDES .shtml

作用是增加处理.shtml 文件的方法。

（38）错误信息的设置。

Alias /error/　" /var/www/error/ "
<IfModule mod_negotiation.c>

```
<IfModule mod_include.c>
    <Directory  " /var/www/error " >
        AllowOverride None
        Options IncludesNoExec
        AddOutputFilter Includes html
        AddHandler type-map var
        Order allow,deny
        Allow from all
        LanguagePriority en es de fr
        ForceLanguagePriority Prefer Fallback
    </Directory>
#   ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
    …
#   ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var
</IfModule>
</IfModule>
```

定义错误编号与文档的对应关系。

（39）为特定的浏览器设置特别的参数。

```
BrowserMatch  " Mozilla/2 "  nokeepalive
…
BrowserMatch  " ^Dreamweaver-WebDAV-SCM1 "  redirect-carefully
```

BrowserMatch 命令为特定的客户程序设置特别的参数，以保证对老版本浏览器的兼容性，并支持新浏览器的新特征。

（40）定义服务器状态和信息。

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from .your-domain.com
</Location>
```

设置 Apache 可以报告详细的服务器状态信息。此配置段与 ExtendedStatus On 相关，一般情况下二者同时起作用。

```
<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from .example.com
</Location>
```

设置 Apache 可以报告详细的服务器配置信息。

（41）代理服务配置模块。

```
<IfModule mod_proxy.c>
        ProxyRequests On
        <Proxy *>
                Order deny,allow
                Deny from all
                Allow from .example.com
        </Proxy>
ProxyVia On
        <IfModule mod_disk_cache.c>
             CacheEnable disk /
             CacheRoot  " /var/cache/mod_proxy "
        </IfModule>
</IfModule>
```

Apache 可以被配置为正向和反向代理。正向代理是一个位于客户端和原始服务器之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。反向代理正好相反，对于客户端而言它就像原始服务器，并且客户端不需要进行任何特别的设置。客户端向反向代理的名字空间中的内容发送普通请求，接着反向代理将判断向何处(原始服务器)转交请求，并将获得的内容返回客户端，就像这些内容原本就是自己的一样。正向代理的典型用途是为防火墙内的局域网客户端提供访问 Internet 的途径。反向代理的典型用途是将防火墙后面的服务器提供给 Internet 用户访问。

3．虚拟主机配置指令

在默认配置文件的最后是一个关于配置虚拟主机的模板，结构如下所示。

```
<VirtualHost 211.67.112.3:80>
       ServerAdmin admin@hncj.edu.cn
       DocumentRoot /www/docs/www.hncj.edu.cn
       ServerName www.hncj.edu.cn
       ErrorLog logs/www.hncj.edu.cn-error_log
       CustomLog logs/www.hncj.edu.cn-access_log common
</VirtualHost>
```

这里由 VirtualHost 来定义虚拟主机，ServerAdmin 用来指定管理员信箱，DocumentRoot 用来指定网站根目录，ErrorLog 用来指定虚拟主机的错误日志文件，CustomLog 用来指定自定义日志文件，其他没有定义的参数使用其前面的定义。

几乎所有虚拟主机配置都有类似的结构，用户可参考此部分内容定义自己的虚拟主机，并且也都放在配置文件的尾部，以使用它们前面的默认配置。

15.4　Web 服务器配置实例

配置 Apache 服务器可以通过几种不同的方法来完成，最有效的方法是直接编辑基本的配置文件，在 Fedora 下还可使用图形配置工具。

15.4.1 Web 服务器配置实例

1．主机访问控制

Apache 的主机访问控制可以用来限制哪些主机可以访问指定的目录。对目录的访问控制主要是通过<Directory>和</Directory>结构实现的。在这里 Order 定义了 allow 与 deny 指令的执行顺序，如果二者有矛盾的话，以最后一次执行的指令为准。需要注意的是 allow、deny 指令的执行顺序与后面的 Allow from 和 Deny from 语句的书写顺序无关。

示例 1：

对于“/var/www/www.hncj.edu.cn”目录做一下访问控制，要求 211.67.113.0/24 的客户机除了 211.67.113.18 外都可以访问，具体配置如下：

```
<Directory " /var/www/www.hncj.edu.cn " >
    Order allow,deny
    Allow from 211.67.113.0/24
    Deny from 211.67.113.18
</Directory>
```

示例 2：

```
<Directory " /var/www/www.hncj.edu.cn " >
    Order deny,allow
    Allow from 211.67.113.0/24
    Deny from 211.67.113.18
</Directory>
```

在这个例子中，实际上 Deny from 211.67.113.18 语句没有起到作用，最后结果是允许 211.67.113.0/24 段所有的客户机都能访问目录“/var/www/www.hncj.edu.cn”。

示例 3：

```
<Directory " /var/www/www.hncj.edu.cn " >
    Order allow,deny
    Allow from all
    Deny from all
</Directory>
```

在这个例子中，只有 Deny from all 真正起到了作用，于是所有的主机都被拒绝访问了。

示例 4：

```
<Directory " /var/www/www.hncj.edu.cn " >
    Order deny,allow
    Allow from 211.67.113.18
    Deny from 211.67.113.18
</Directory>
```

请读者考虑 211.67.113.18 这个客户机是否被允许访问。

关于主机访问控制有以下几点说明：

（1）Order 命令中指定的顺序至关重要，写在后面的关键字有最终决定权。

（2）IP 地址的表示方法比较灵活，可以有如下写法。

① 网络/子网掩码：211.67.113.0/255.255.255.0

② CIDR 规范：211.67.113.0/24

③ 单个 IP 地址的表示：211.67.113.0

（3）也可使用域名，如 Allow from hncj.edu.cn。

2．虚拟主机

下面介绍这三种虚拟主机的创建方法。

1）默认虚拟主机

要求：在同一台主机上针对一个 IP 地址和不同的端口来建立虚拟主机，即每个端口对应一个虚拟主机。

具体实现步骤如下：

步骤 1：为 eth0 创建网络子接口 eth0:1，并给这个接口指定 IP 地址（具体的 IP 地址视服务器而定）。

```
#ifconfig eth0:1 192.168.64.161 netmask 255.255.255.0
```

步骤 2：编辑 Apache 的主配置文件。

编辑 httpd.conf，增加监听的端口号 8080。在 Section1 的 Listen 80 下增加下面的行：

```
Listen 8080
```

步骤 3：在主配置文件末尾加入以下内容：

```
<VirtualHost 192.168.64.161:8080>
    ServerAdmin admin@hncj.edu.cn
    DocumentRoot /var/www/www8080
    ServerName 192.168.64.161
    ErrorLog logs/192.168.64.161-8080-error_log
    CustomLog logs/192.168.64.161-8080-access_log common
</VirtualHost>
```

在这里，利用<VirtualHost 192.168.64.161:8080>来定义基于自定义端口的虚拟主机。

步骤 4：建立虚拟主机的文档根目录及主页。

（1）创建虚拟主机文档目录。

```
#mkdir /var/www/www8080                    #若已有可不创建
```

（2）编辑生成主页文件 index.html。

编辑/var/www/www8080/index.html 文件，在 index.html 中加入以下内容：

```
<html>
     <body>Default Virtual Host test,Port 8080.</body>
</html>
```

步骤 5：运行与测试。

```
#systemctl reload httpd
#elinks http://192.168.64.161:8080
```

2）基于 IP 地址的虚拟主机

要求：在同一主机上配置多个 IP 地址，每个 IP 地址对应一个虚拟主机。

步骤 1　为 eth0 创建子接口 eth0:2 和 eth0:3，并分别配置 IP 地址 192.168.64.162 和 192.168.64.163

```
#ifconfig eth0:2 192.168.64.162 netmask 255.255.255.0
#ifconfig eth0:3 192.168.64.163 netmask 255.255.255.0
```

步骤 2　编辑 Apache 主配置文件 httpd.conf。

编辑主配置文件 httpd.conf，在文件的末尾加入以下内容：

```
<VirtualHost 192.168.64.162:80>
    ServerAdmin admin@hncj.edu.cn
    DocumentRoot /var/www/www162
    ServerName 192.168.64.162
    ErrorLog logs/192.168.64.162-error_log
    CustomLog logs/192.168.64.162-access_log common
</VirtualHost>
<VirtualHost 192.168.64.163:80>
    ServerAdmin admin@hncj.edu.cn
    DocumentRoot /var/www/www163
    ServerName 192.168.64.163
    ErrorLog logs/192.168.64.163-error_log
    CustomLog logs/192.168.64.163-access_log common
</VirtualHost>
```

这里需要注意的是，除了 VirtualHost 要指定对应的 IP 地址外，还要在 ServerName 指定，这一点很重要。

步骤 3　建立两个虚拟主机的文档目录及相应测试页面。

```
#mkdir /var/www/www162
#mkdir /var/www/www163
```

分别编辑 www162 和 www163 站点的默认主页文档，内容为：

```
#/var/www/www162/index.html 的内容
     <html>
          <body>IP based Virtual Host test,IP address 192.168.64.162.</body>
     </html>
#/var/www/www163/index.html 内容
     <html>
          <body>IP based Virtual Host test,IP address 192.168.64.163.</body>
     </html>
```

步骤 4　运行与测试。

```
#systemctl reload httpd
#elinks http://192.168.64.162
#elinks http://192.168.64.163
```

3）基于名字的虚拟主机

因服务器数量不足、IP 地址短缺等各方面的原因，很多站点要部署在一台服务器上，甚至要共享一个 IP 地址。虽然可以通过基于端口虚拟主机的方法为每个虚拟主机建立不同的端口，但当用户要访问不同的站点时，要在站点 IP 地址或域名后加入不同端口号，既不方便用户访问，又会影响站点的推广。为了解决这些问题，基于名字的虚拟主机应运而生，并得到广泛的应用。

要求：在同一主机上针对相同的 IP 地址和端口号建立基于名字的虚拟主机。

步骤 1　在 DNS 服务器的区域数据库文件中增加两条 A 记录（默认域名为 hncj.edu.cn，或结合实际需要），并重载配置。

```
tmgc IN    A     192.168.64.164
szgc  IN    A     192.168.64.164
```

步骤 2　为 Apache 服务器增加子接口 eth0:4，配置 IP 地址 192.168.64.164。

```
#ifconfig eth0:2 192.168.64.164 netmask 255.255.255.0
```

步骤 3　编辑 Apache 服务主配置文件 httpd.conf，在文件末尾加入以下内容：

```
NameVirtualHost 192.168.64.164:80
<VirtualHost 192.168.64.164:80>
      ServerAdmin admin@hncj.edu.cn
      DocumentRoot /var/www/tmgc
      ServerName tmgc.hncj.edu.cn
      ErrorLog logs/tmgc.hncj.edu.cn -error_log
      CustomLog logs/tmgc.hncj.edu.cn -access_log common
</VirtualHost>
<VirtualHost 192.168.64.164:80>
      ServerAdmin admin@hncj.edu.cn
      DocumentRoot /var/www/szgc
      ServerName szgc.hncj.edu.cn
      ErrorLog logs/szgc.hncj.edu.cn-error_log
      CustomLog logs/szgc.hncj.edu.cn -access_log common
</VirtualHost>
```

NameVirtualHost 是一条非常重要的指令，只有在该指令的作用下才能激活 192.168.64.164 上基于名字虚拟主机的功能。两个基于名字的虚拟主机主要的区别在于 ServerName 值，前面两种类型的虚拟主机采用的都是 IP 地址，而基于名字的虚拟主机只能采用域名。配置完毕后，用户就只能根据域名来访问两个站点。

15.4.2　Fedora 的图形配置工具

Fedora 下还保留有 Apache 服务器的图形配置工具，用户借助此工具可以帮助对配置文件内容的理解。

1．Apache 服务器 httpd 图形配置工具的安装与启动

Fedora 的 Apache 图形配置工具软件包为 system-config-httpd，安装方法是：

```
#dnf install system-config-httpd
```

启动方法是按顺序选择“Applications（应用程序）”→“System Tools（系统工具）”→“HTTP”，或从 CLI 界面执行命令：

```
#system-config-httpd
```

启动后进入图形设置界面（图略）。

2．Apache 图形配置工具的使用

（1）Main（主选项卡）

在该选项卡中可以设置服务名称、管理员邮箱地址、服务器地址和端口等信息。

（2）Virtual Hosts（虚拟主机）选项卡

用 Apache 实现的虚拟主机主要有 3 种类型：默认虚拟主机（Default Virtual Host，也称基于端口的虚拟主机）、基于 IP 地址的虚拟主机（IP based Virtual Host）和基于名字的虚拟主机（Name based Virtual Host），其中基于名字的虚拟主机需要与 DNS 服务器进行配合才能正常使用。在这个选项卡中不但可以添加基于 IP 地址的虚拟主机、基于端口的虚拟主机和基于域名的虚拟主机，还可以对这些虚拟主机进行详细的设置，同时还可以设置默认虚拟主机配置。

（3）Server（服务器）选项卡

Server 选项卡用于配置 User 和 Group 和 PidFile 等参数。

（4）Performance Tuning（性能调整）选项卡

用于配置与性能有关的参数，如 MaxClients，TimeOut，KeepAliveTimeout 和 MaxRequestPerChild 等参数。

使用此版本的 system-config-httpd 工具配置时，并不修改原配置文件，而是将自己产生的配置文件命名为 system-config-httpd.conf，存放在目录/etc/httpd/conf.d 中，在 httpd 启动时被自动读入（参见 Include conf.d/*.conf）。

15.5　httpd 与防火墙和 SELinux 的关系

通过命令 man httpd_selinux 可以获得 SELinux 与 httpd 关系的详尽信息。

15.5.1　httpd 与防火墙的关系

1．Firewalld 防火墙

可以选择“Applications（应用程序）”→“Sundry（杂项）”→“Firewall”打开防火墙配置图形界面，在使用域上勾选 http 并使之生效。

或以超级用户身份执行以下命令：

```
#firewall-cmd --add-service=http                    #增加允许 http 的规则
```

2．Ubuntu 的 ufw 防火墙

在 Ubuntu 系统中，若使用的是 ufw 防火墙，可在 CLI 界面执行命令：

```
$ sudo ufw allow http                    #增加允许 http 的规则
```

或以超级用户身份直接执行命令：

```
#ufw allow http                    #增加允许 http 的规则
```

15.5.2　httpd 与 SELinux 的关系

1．布尔变量

可以使用命令

```
#getsebool -a | grep httpd
```

得到 SELinux 与 httpd 相关的布尔变量及其值。常见的布尔变量及意义如下：

httpd_builtin_scripting：是否允许 httpd 支持 built-in scripting；

httpd_can_network_connect：是否允许 httpd 脚本和模块链接网络；

httpd_can_network_connect_db：是否允许 httpd 脚本和模块链接数据库；

httpd_can_network_relay：是否允许 httpd 作为中继；

httpd_enable_cgi：是否允许 httpd 支持 cgi；

httpd_enable_ftp_server：是否允许 httpd 作为 ftp 服务器；

httpd_enable_homedirs：是否允许 httpd 读家目录；

httpd_graceful_shutdown：是否允许 httpd 通过端口 80 进行优雅关闭；

httpd_ssi_exec：是否允许 httpd 执行 ssi 元素；

httpd_tty_comm：是否允许 httpd 访问控制终端；

httpd_unified：是否允许 httpd_t 完全访问 httpd 类型；

httpd_use_cifs：是否允许 httpd 访问 samba/cifs 文件系统；

httpd_use_nfs：是否允许 httpd 访问 nfs 文件系统。

SELinux 对 httpd 所使用的控制变量如下，可用于控制 httpd 的行为。httpd 服务器安装时，系统自动打开了 httpd_builtin_scripting、httpd_enable_cgi 和 httpd_graceful_shutdown 布尔变量，

需要时还可以打开其他布尔变量，方法是：

```
#setsebool [-P] bool_var on          #使用-P 时永久打开，否则临时打开，系统重启后失效
```

2．类型标签

在 SELinux 中，与 httpd 相关的文件和目录标签如表 10-9 所示。

3．httpd 与 SELinux 的关系

在默认情况下，在 SELinux 的控制下，httpd 服务器以限定方式运行，工作在自己域内。如果此时 httpd 被攻破，攻击所造成的危害是相当小的。在 SELinux 的控制下，httpd 进程的状态可由#ps -eZ | grep httpd 来查询，命令的输出为：

```
unconfined_u:system_r:httpd_t:s0 8179 ?          00:00:00 httpd
```

由此可见，SELinux 与 httpd 相关的上下文为 unconfined_u:system_r:httpd_t:s0，httpd_t 为其类型标签。httpd 默认的监听端口可以通过命令

```
#semanage port -l|grep http_port_t
        http_port_t              tcp          80, 443, 488, 8008, 8009, 8443
```

来查询，如果要改变增加默认监听端口（PORT），则需要通过

```
#semanage port -a -t http_port_t -p tcp PORT
```

来添加。

SELinux 的策略决定了 httpd 如何在自己域中运行以及如何与其他进程和文件的互动关系，相关文件的标签必须设置正确，比如具有 httpd_sys_content_t 标签的文件只能被 httpd 读取而不能写入，哪怕是操作系统允许也不行。

习题 15

1．简述 Apache 的特点。
2．Apache 的配置文件分几节，作用分别是什么？
3．Apache 服务器能创建的虚拟主机有几种，优缺点是什么？

实验 15

1．在 Linux 系统中，使用 Apache 建立自己的个人网站。
2．建立 3 种不同的虚拟主机。

第 16 章　DHCP 服务器

16.1　DHCP 介绍

16.1.1　DHCP 协议

DHCP 是 Dynamic Host Configuration Protocol（动态主机分配协议）的缩写，它的前身是 BOOTP。BOOTP 原本是用于无磁盘主机连接的网络环境中，网络主机使用 BOOT ROM 而不是磁盘启动并连接上网络，通过 BOOTP 自动为那些主机设定 TCP/IP 环境。但 BOOTP 有一个缺点，是在设定前需事先获得客户端的硬件地址，而且 IP 是静态的。换而言之，BOOTP 缺乏“动态性”，若在有限的 IP 资源环境中，BOOTP 的一对一会造成非常严重的资源浪费。

DHCP 是 BOOTP 的增强版本，它分为两部分：一个是服务器端，而另一个是客户端。在此机制下，DHCP 服务器集中管理所有的 IP 网络设定数据，并负责处理客户端的 DHCP 要求；而客户端则会使用从服务器分配下来的 IP 环境数据，配置自己的网络参数。每个 DHCP 客户连接到 DHCP 服务器上时，服务器就会返回 IP 地址、网关和 DNS 服务器等信息给客户端。让其以此配置自己的网络参数。相对 BOOTP，DHCP 通过“租约”的概念，有效且动态地分配客户端的TCP/IP 相关属性的设定，而且作为兼容考虑，DHCP 也完全照顾了 BOOTP Client 的需求。

DHCP 详尽的协议内容在 RFC 文档的 RFC2131 和 RFC1541 中有描述。

1．DHCP 管理的两种基本数据

DHCP 管理的两种基本数据如下所述。

（1）地址池中的地址：是可以由 DHCP 动态分配的可用 IP 地址。

（2）租用地址：是 DHCP 已经从地址中分配出的地址。

下面是几个与此相关的基本概念。

（1）DHCP 客户：是指通过 DHCP 服务器来获得网络配置参数的主机。

（2）DHCP 服务器：是指提供网络配置参数给 DHCP 客户机的主机。

（3）DHCP 中继代理：是指 DHCP 服务器和 DHCP 客户之间转发 DHCP 消息的主机或路由器。

（4）租用：是指 DHCP 客户从 DHCP 服务器上获得并临时占用该 IP 地址的过程。

2．DHCP 提供两种 IP 分配方式

（1）自动分配（Automatic Allocation）：DHCP 客户端第一次成功地从 DHCP 服务器端租用到 IP 地址之后就永远使用这个地址。

（2）动态分配（Dynamic Allocation）：当 DHCP 客户第一次从 HDCP 服务器端租用到 IP 地址之后并非永久使用该地址，只要租约到期客户端就得释放这个 IP 地址，以便给其他工作站使用。当然，客户端也可以延续租约或是租用其他的 IP 地址。

DHCP 除了能动态地设定 IP 地址之外，还可以将一些 IP 保留下来给一些特殊用途的机

器使用，也可以按照 MAC 地址来分配固定的 IP 地址，这样可以带来更大的设计空间。

16.1.2　DHCP 的工作过程

根据客户端是否是第一次登录网络，DHCP 的工作形式会有所不同。

第一次登录时：当 DHCP 客户端第一次登录网络的时候，也就是客户发现本机上没有任何 IP数据设定，它会向网络发出一个 DHCPDISCOVER 数据包。因为客户端还不知道自己属于哪一个网络，所以数据包的源地址标记为 0.0.0.0，而目的地址则为 255.255.255.255，然后再附上 DHCPDISCOVER 的信息，向网络进行广播。

预设的 DHCPDISCOVER 等待时间为 1s，也就是当客户端将第一个 DHCPDISCOVER 数据包送出去之后，在 1s 之内没有得到响应的话，就会进行第二次 DHCPDISCOVER 广播。若一直得不到响应，客户端会有四次 DHCPDISCOVER 广播（包括第一次在内），除了第一次会等待 1s 之外，其余三次的等待时间分别是 9s、13s、16s。如果都没有得到 DHCP 服务器的响应，客户端则会显示错误信息，宣告 DHCPDISCOVER 的失败。之后，基于使用者的选择，系统会继续在 5min 之后再重复一次 DHCPDISCOVER 的过程。

提供 IP 租用地址：当 DHCP 服务器监听到客户端发出的 DHCPDISCOVER 广播后，它会从那些还没有租出的地址范围内选择最前面的空闲 IP 地址，连同其他 TCP/IP 设定，回应给客户端一个 DHCPOFFER 数据包。由于客户端在开始的时候还没有 IP 地址，所以在其 DHCPDISCOVER 数据包内还会带有 MAC 地址信息，并且有一个 XID 编号来辨别该数据包，DHCP 服务器响应的 DHCPOFFER 数据包则会根据这些资料传递给要求租约的客户。根据服务器端的设定，DHCPOFFER 数据包会包含一个租约期限的信息。

接受 IP 租约：如果客户端收到网络上多台 DHCP 服务器的响应，只会挑选其中一个 DHCPOFFER（通常是最先抵达的那个），并且会向网络发送一个 DHCPREQUEST 广播数据包，告诉所有 DHCP 服务器它将指定接收哪一台服务器提供的 IP 地址。同时客户端还会向网络发送一个 ARP 封包，查询网络上面有没有其他机器使用该 IP 地址；如果发现该 IP 已经被占用，客户端则会送出一个 DHCPDECLINE 数据包给 DHCP 服务器，拒绝接收其 DHCP OFFER，并重新发送 DHCPDISCOVER 信息。

租约确认：当 DHCP 服务器接收到客户端的 DHCPREQUEST 之后，会向客户端发出一个 DHCPACK 响应，以确认 IP 租约的正式生效，也就结束了一个完整的 DHCP 工作过程。具体的工作流程如图 16-1 所示。

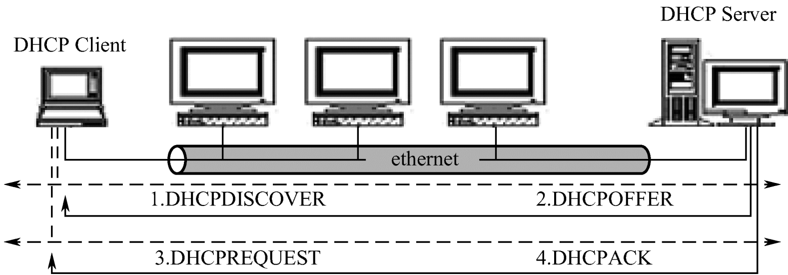


图 16-1　DHCP 的工作流程

客户机取得 IP 租约后，即可使用该 IP 地址实现网络服务。若采用“动态分配”的方式，

每次登录网络时都需要通过以上的步骤获取；而如果采用“自动分配”的方式，DHCP 客户机必须定期更新租约，否则当租期到期时，就不能再使用此 IP 地址。按照 RFC 的默认规定，每当租用时间超过租约的 50%或 87.5%时，客户机就必须发出 DHCPREQUEST 信息包，向 DHCP 服务器请求更新租约。在更新租约时，DHCP 客户机是以单点传送方式发出 DHCPREQUEST 信息包，不再进行广播。具体过程如下：

（1）在租期超过 50%时，DHCP 客户机直接向为其提供 IP 地址的 DHCP 服务器发送 DHCPREQUEST 消息包，如果接到服务器的 DHCPACK 数据包则租期更新完成，如没有收到服务器端发送的 DHCPACK 数据包，仍可继续使用，但要在 87.5%时再次申请更新租约。

（2）在租期超过 87.5%，再次向 DHCP 服务器发送 DHCPACK 数据包，若仍未完成，则当 IP 地址租期到达时应放弃该 IP 地址，重新开始 IP 租用过程。

16.2 DHCP 服务器的安装与启动管理

16.2.1 DHCP 服务器软件安装

红帽系统提供的 DHCP 服务器软件包为 dhcp-server，ubuntu 系统提供的为 isc-dhcp-server，使用前要先安装它，安装方法如下：

```
#yum install dhcp-server                    #红帽系统
#apt install isc-dhcp-server                #ubuntu
```

16.2.2 启动管理

在红帽系统下的 dhcp 服务名为 dhcpd.service 和 dhcpd6.service，在 ubuntu 下的服务名为 isc-dhcp-server.service 和 isc-dhcp-server6.service，用户可用服务管理的方法对其进行管理。这里，我们只关心 IPv4 的 DHCP 服务。

1．检查服务状态

```
systemctl status dhcpd.service                         #红帽系统
systemctl status isc-dhcp-server.service               #ubuntu 系统
```

2．启用服务或设为开机重启动

```
systemctl enable dhcpd.service                         #红帽系统
systemctl enable isc-dhcp-server.service               #ubuntu
```

3．启动、关闭或重启动服务

```
systemctl start/stop/restart dhcpd.service             #红帽系统
systemctl start/stop/restart isc-dhcp-server.service   #ubuntu
```

16.3 DHCP 的配置

16.3.1 配置文件

Fedora24 中的 dhcpd 的配置文件是/etc/dhcpd/dhcpd.conf。默认情况下此文件不存在，或是一个只有注释内容的空文件，其内容为：

```
##DHCP Server Configuration file.
#see /usr/share/doc/dhcp-server/dhcpd.conf.example
#see dhcpd.conf(5) man page
```

要求用户参见/usr/share/doc/dhcp-server/dhcpd.conf.example，或 man 5 dhcpd.conf。

事实上，当安装了 dhcp-server 后，系统就提供了一个配置模版文件/usr/share/doc/dhcp-server/dhcpd.conf.example。首次使用 DHCP 时，可将它复制成/etc/dhcpd/dhcpd.conf，然后再根据实际需要对其进行修改，现就其内容注释如下。

```
#dhcpd.conf
#Sample configuration file for ISC dhcpd
#option definitions common to all supported networks...
option domain-name "example.org";                    #为 DHCP 客户设置 DNS 域
option domain-name-servers ns1.example.org, ns2.example.org; #为 DHCP 客户设置 DNS 服务器地址
default-lease-time 600;                              #为 DHCP 客户设置默认的地址租期
max-lease-time 7200;                                 #为 DHCP 客户设置最长的地址租期
#ddns-update-style none;                             #配置 DHCP-DNS  互动更新模式
#authoritative;                                      #设置为权威或官方
log-facility local7;                                 #定义日志记录位置

subnet 10.152.187.0 netmask 255.255.255.0 {          #设置子网结构
}

#定义最基本子网
subnet 10.254.239.0 netmask 255.255.255.224 {
  range 10.254.239.10 10.254.239.20;                 #地址范围
  option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;     #为客户端设置默认网关
}

#为 BOOTP 客户端定义动态子网（但不推荐）
subnet 10.254.239.32 netmask 255.255.255.224 {
  range dynamic-bootp 10.254.239.40 10.254.239.60;   #地址范围
  option broadcast-address 10.254.239.31;            #广播地址
  option routers rtr-239-32-1.example.org;           #默认网关
}
```

```
#定义一个内部子网
subnet 10.5.5.0 netmask 255.255.255.224 {             #定义一个内部子网
  range 10.5.5.26 10.5.5.30;                           #地址范围
  option domain-name-servers ns1.internal.example.org; #DNS
  option domain-name "internal.example.org";           #所在域名
  option routers 10.5.5.1;                             #路由器
  option broadcast-address 10.5.5.31;                  #广播地址
  default-lease-time 600;                              #默认地址租期
  max-lease-time 7200;                                 #最长地址租期
}
#定义有特殊设置的主机
#有特殊要求主机的特殊设置，应该写在 host 语句中，若没有指定地址的话将会自动分配
#但是特殊设置将来自本 host 定义
host passacaglia {                                     #定义特殊设置的主机
  hardware ethernet 0:0:c0:5d:bd:95;                   #客户的 MAC 地址
  filename "vmunix.passacaglia";                       #指定客户端启动要载入的初始启动文件
  server-name "toccata.fugue.com";                     #服务器主机名
}

#定义固定 IP 的主机，具有固定地址的主机可以从 BOOTP 或 DHCP 启动
#但具有非固定地址的主机只能从 DHCP 启动，除非有某种特殊情况
host fantasia {
  hardware ethernet 08:00:07:26:c0:a5;                 #客户端 MAC
  fixed-address fantasia.fugue.com;                    #对此 MAC 分配固定的 IP 地址
}

#定义一个类 foo，然后可以基于此类进行地址分配
#示例将说明某个子网上的所有主机从 10.17.224/24 获得地址
#而其他的主机将从 10.0.29/24 获得地址
class "foo" {                                          #定义类，符合以下条件
  match if substring (option vendor-class-identifier, 0, 4) = "SUNW";
}

shared-network 224-29 {                                #共享 10.17.224/24 者
  subnet 10.17.224.0 netmask 255.255.255.0 {
    option routers rtr-224.example.org;
  }
  subnet 10.0.29.0 netmask 255.255.255.0 {             #其他使用 10.0.29/24 者
    option routers rtr-29.example.org;
  }
  pool {        #地址池
    allow members of "foo";                            #允许 foo 类中成员
    range 10.17.224.10 10.17.224.250;
  }
  pool {                                               #地址池
```

```
        deny members of "foo";                    #拒绝 foo 类中成员
        range 10.0.29.10 10.0.29.230;
      }
    }
```

16.3.2　配置文件中的定义、参数和选项及意义

1．定义

定义也叫声明，用于定义网络布局、提供给客户端的 IP 地址等，表 16-1 给出了常用的定义及其解释。

表 16-1　常用的定义及其解释

定　　义	功　　能
shared-network 名称 { ... }	定义超级作用域
subnet 网络号 netmask 子网掩码 { ... }	定义作用域（或 IP 子网）
range 起始 IP 地址 终止 IP 地址	定义作用域（或 IP 子网）范围
host 主机名 { ... }	定义特殊设置的主机
group { ... }	定义一组参数
Pool { … }	定义地址池

2．参数

参数是可选的或控制 DHCP 服务器行为的值，表 16-2 给出了常用的参数及其解释。

表 16-2　常用的参数及其解释

参　　数	功　　能
ddns-update-style 类型	定义所支持的 DNS 动态更新类型（可选项有 ad-hoc、interim 和 none），只用于全局
allow/ignore client-updates	允许/忽略客户机更新 DNS 记录 只用于全局
default-lease-time 数字	指定默认的租约期限
max-lease-time 数字	指定最大租约期限
hardware 硬件类型 MAC 地址	指定网卡接口类型和 MAC 地址
server-name 主机名	通知 DHCP 客户机服务器的主机名
fixed-address ip 地址	分配给客户端一个固定的 IP 地址

3．选项

选项是用来配置 DHCP 客户端的可选参数，它们全部用 option 关键字作为开头，表 16-3 给出了常用的选项及解释。

表 16-3　常用的选项及其解释

选　　项	功　　能
subnet-mask 子网掩码	为客户端指定子网掩码
domain-name “域名”	为客户端指定域名
domain-name-server ip 地址	为客户端指定 DNS 服务器的 IP 地址
host-name “主机名”	为客户端指定主机名
routers ip 地址	为客户端指定默认网关
broadcast-address 广播地址	为客户端指定广播地址
netbios-name-servers ip 地址	为客户端指定 WINS 服务器的 IP 地址
netbios-node-type 节点类型	为客户端指定节点类型
ntp-server ip 地址	为客户端指定网络时间服务器的 IP 地址
nis-servers ip 地址	为客户端指定 NIS 域服务器的 IP 地址
nis-domain “名称”	为客户端指定所属 NIS 域的名称
time-offset 偏移量	为客户端指定与格林尼治时间的偏移量或时区

4．配置举例

下面是此文件的一个例子。

```
ddns-update-style interim;                    #配置使用过渡性 DHCP-DNS 互动更新模式
ignore client-updates;                        #忽略客户端更新
subnet 192.168.1.0 netmask 255.255.255.0 {    #设置子网
    range 192.168.1.10 192.168.1.100;         #设置地址池的可用地址范围
    range 192.168.1.150 192.168.1.200;
    option routers         192.168.1.254;     #设置默认网关
    option subnet-mask    255.255.255.0;      #设置子网掩码
    option broadcast-address 192.168.1.255;   #设置广播地址
    option domain-name     "mydomain.org";    #设置域名字
    option domain-name-servers     192.168.1.1,192.168.1.2; #设置 DNS 地址
    default-lease-time             1200;      #设置默认的地址租期
    max-lease-time                 9200;      #设置最长的地址租期
    host pc1 {
        hardware Ethernet 12:34:56:78:AB:CD;  #为该 MAC 主机
        fixed-address 192.168.1.20;           #保留特定的 IP 地址
    }
}
```

在了解相关的定义、参数和选项的含义之后，用户可以根据实际情况来定制自己的 dhcpd.conf 文件内容。

16.4 DHCP 规划

16.4.1 在不同的网络中使用 DHCP

DHCP 作为一种机制，它必须允许本地管理员控制配置参数，使客户不需要手工配置就可以获取网络配置参数，同时还能够使每一台连接到网络中的计算机自动获得唯一的动态 IP 地址，并且能够为早期的 BOOTP 客户提供兼容的服务。

由于对 DHCP 服务器可以服务的客户端最大数量和可以在 DHCP 服务器上创建的作用域数量没有固定限制，因此在确定要使用的 DHCP 服务器数目时，最主要的考虑因素是网络体系结构和服务器硬件。在组织网络中部署 DHCP 服务器前，可以先对它进行测试，以确定硬件的限制和性能并了解网络体系结构、通信和其他因素是否影响 DHCP 服务器的性能。通过硬件和配置测试，还可以确定每台服务器要配置的作用域数量。

1．小型网络中 DHCP 的部署

在只有一个子网的小型网络中，可使用单一的 DHCP 服务器为网络中的主机提供服务，应考虑以下因素：

（1）网络中提供特殊服务（DNS、FTP）的服务器应该采用静态 IP 地址。

（2）不同的网络服务应该由不同的服务器承担，此方法可以做到在某个网络服务停止时其他的服务器仍然可以正常工作，以此来提高网络的可用性和安全性。

2．在大型网络中部署 DHCP

在大型网络中，由于主机数目较多、网络结构相对复杂、网络应用多、网络的安全性要求较高等原因，使得 DHCP 的部署更为复杂。在网络中物理子网划分的数量以及路由器的位置将对 DHCP 性能发挥的影响非常明显，因为广播包在物理子网之间的传输和对路由器的跨越是比较困难的。

在大型网络中通常会采用路由器将网络划分为多个物理子网，而路由器的主要功能就是屏蔽子网间的广播，减少网络带宽占用，提高网络性能。如果子网中没有一台 DHCP 服务器，那么每个子网中的主机将没有办法跨越子网来定位 DHCP 服务器，也就无法获得 IP 地址。要想解决这样的问题，可以有两种方法：

（1）为每一个子网配置一台 DHCP 服务器，可以有效地解决每个子网中的主机方便地获取 IP 地址，但是该方法的不足之处是网络管理将会有一定的麻烦。

（2）在每个子网中设定一个 DHCP 中继代理。网络中的主机将 IP 请求发送给中继代理，由中继代理向 DHCP 服务器发出请求，并将获得的 IP 地址返回 DHCP 客户机。

16.4.2 设置 DHCP 中继

DHCP 中继代理允许将无 DHCP 服务器子网内的 DHCP 客户机的 DHCP 请求转发给其他子网的一个或多个 DHCP 服务器。当某个 DHCP 客户机发出请求信息时，DHCP 中继代理把

该请求转发给 DHCP 中继代理启动时所设定的 DHCP 服务器，当 DHCP 服务器返回应答时，该应答被广播或单播给发出请求的客户机所在的网络。需要注意的是 DHCP 中继代理本身应该具有静态 IP 地址。

若要使用 DHCP 中继，需要安装 DHCP 中继及相关软件包。在红帽系列系统下，该软件包为 dhcp-relay，在 ubuntu 下该软件包为 isc-dhcp-relay，它们所对应的服务分别为 dhcrelay.service 和 isc-dhcp-relay.service（isc-dhcp-relay6.service）。

软件包的安装方法是：

```
yum install dhcp-relay                          #红帽系统
apt install isc-dhcp-relay                      #ubuntu 系统
```

DHCP 中继代理启动管理方法如下：

1．检查服务状态

```
systemctl status dhcrelay.service               #红帽系统
systemctl status isc-dhcp-relay.service         #ubuntu 系统
```

2．启用服务或设为开机重启动

```
systemctl enable dhcrelay.service               #红帽系统
systemctl enable isc-dhcp-relay.service         #ubuntu
```

3．启动、关闭或重启动服务

```
systemctl start/stop/restart dhcrelay.service           #红帽系统
systemctl start/stop/restart isc-dhcp-relay.service     #ubuntu
```

16.4.3 设置备份 DHCP

在一个具有大量主机和子网的大型网络中，DHCP 的冗余是一个非常关键的问题。由于 DHCP 服务器承担着为全网主机提供 IP 地址分配的责任，如果不考虑 DHCP 的冗余，只需要再为每个子网中设置一台 DHCP 服务器，也不需要使用 DHCP 中继代理，但要考虑安全性问题，一旦 DHCP 服务器出现故障，则整个网络中的主机都将不能访问网络资源。为解决此问题，可以通过同时配置多个 DHCP 服务器来相互实现冗余，以保证网络的可用性。但由于 DHCP 服务器本身并不提供备份功能，所以容错能力只能通过将不同的 IP 地址段分发到不同的 DHCP 服务器上来实现，让多个不同的 DHCP 服务器为同一个网络提供 DHCP 服务，最可行的解决方案就是在两个子网的 DHCP 服务器上相互为对方的子网提供 DHCP 服务，然后通过设置 DHCP 中继代理或路由器来转发 DHCP 广播包以达到冗余的目的，从而保证 DHCP 服务的可用性和安全性。

例如，以下两个网络（192.168.1.0 和 192.168.2.0）中的 DHCP 服务器的设置：

#子网 192.168.1.0 中的 DHCP 服务器的配置如下：

```
subnet    192.168.1.0       netmask 255.255.255.0 {
     range   192.168.1.10         192.168.1.199;
}
subnet    192.168.2.0       netmask 255.255.255.0 {
```

```
        range   192.168.2.100        192.168.2.199;
    }
```

#子网 192.168.2.0 中的 DHCP 服务器的配置如下：

```
    subnet    192.168.2.0        netmask 255.255.255.0 {
        range   192.168.2.10         192.168.2.199;
    }
    subnet    192.168.1.0        netmask 255.255.255.0 {
        range   192.168.1.100        192.168.1.199;
    }
```

以上两个 DHCP 服务器均能够对两个网络上的客户机提供 DHCP 服务，以实现 DHCP 的冗余。

16.5　DHCP 客户端设置

在 Linux 系统中，网络配置可以理解为 DHCP 客户端设置。关于系统的网络配置，可通过图形界面方式和手工修改配置文件两种方式进行，具体参见第 9 章的网络设置部分。

16.6　DHCP 与防火墙的关系

1．firewalld

在图形界面下，可根据需要在所用域内勾选 dhcp、dhcp6 和 dhcp6-client 等服务，需要时重启 firewalld 防火墙，也可在 CLI 界面下根据需要执行以下命令：

```
#firewall-cmd --add-service=dhcp              #添加 dhcp 服务，其他仿此方法
#systemctl restart firewalld.service          #需要时重启 firewalld 防火墙
```

2．ufw

执行以下命令：

```
#ufw allow boops                              #打开 bootps
#ufw allow bootpc                             #打开 bootpc
```

3．与 SELinux 的关系

SELinux 没有提供对 dhcp 等控制的布尔变量，与 dchp 相关的标签类型参见表 10-9。要保持 dhcp 相关文件的 SELinux 标签正确，若遇到问题，可参考 10.5.7 节进行处理。

习题 16

1．简答题

（1）简述 DHCP 的功能。
（2）简述 DHCP 配置文件的相关选项及含义。
（3）简述在不同网络环境中 DHCP 的规划。
（4）简述 DHCP 中继的含义及其应用。

2．综合题

（1）试述 DHCP 的工作原理。
（2）在设置 DHCP 备份时，应该注意和解决哪些问题？

实验 16

1．完成 DHCP 的安装和配置。
2．配置 DHCP 服务器。

第 17 章　FTP 与 TFTP 服务器

17.1　FTP 与 FTP 服务器概述

在众多网络应用中，FTP（File Transfer Protocol）有着非常重要的地位。20 世纪末，各种各样的资源大多数都是放在 FTP 服务器中的，可以说，FTP 与 Web 服务几乎占据了整个 Internet 应用的 80%以上。目前，虽然随着 P2P 技术的发展，FTP 应用正逐步减少，但是在校园、科研单位等网络中，FTP 仍然是一种相当重要的服务。

17.1.1　FTP 的相关概念

1．FTP 协议和 FTP 服务

FTP 是 File Transfer Protocol（文件传输协议）的英文简称，是 TCP/IP 协议族中的一种具体应用，用于 Internet 上控制文件的双向传输。该协议是 Internet 文件传送的基础，文件传输协议标准是在 RFC959 中说明的，该协议定义了一个远程计算机系统和本地计算机系统之间传输文件的标准，它的目标是提高文件的共享性。简单地说，FTP 就是要完成两台计算机之间的复制，从远程计算机复制文件至本地计算机，称之为“下载”文件。若将文件从本地计算机复制至远程计算机，则称之为“上传”。

FTP 协议工作在 OSI 模型的第七层（应用层）上，并利用传输控制协议 TCP 在不同的主机之间提供可靠的数据传输。TCP 是一种面向连接的、可靠的传输协议，正是这种可靠性保证了 FTP 文件传输的可靠性，也就是说在客户端与服务器之间传输数据包时，一旦出现丢包现象，就会重新传输以保证数据的正确性。FTP 要使用两个 TCP 连接工作，一个是命令连接，用来在 FTP 客户端和服务器之间传递命令，另一个是数据连接，用来传送数据。这就直接导致 FTP 需要两个端口，一个端口用于控制连接，即 21 号端口给服务器发送命令以及等待服务器响应；另一个是数据传输端口，端口号是 20，用来建立数据传输通道。

FTP 服务的一个重要特点是其实现可以独立于平台，不同的操作系统都有自己的 FTP，因而可以实现文件信息在不同的平台间传输。

2．FTP 的数据传输模式

根据数据传输连接建立方式的不同，可以把 FTP 分成主动模式（Active）和被动模式（Passive）。

（1）主动模式：主动模式也称 PORT 模式。在此模式下，FTP 客户端随机开启一个大于 1024 的端口 *N* 向服务器的 21 号端口发起连接，然后开放 *N*+1 号端口进行监听，并向服务器发出 PORT *N*+1 命令。服务器接收到命令后，会用其本地的 FTP 数据端口（通常是 20）来连接客户端指定的端口 *N*+1，进行数据传输。在主动模式下，FTP 的数据连接和控制连接的方向是相反的，也就是说，是服务器向客户端发起一个用于数据传输的连接。客户端的连接端口是由服务器和客户端通过协商确定的。

（2）被动模式：FTP 客户端随机开启一个大于 1024 的端口 *N* 向服务器的 21 号端口发起连接，同时会开启 *N*+1 号端口。然后向服务器发送 PASV 命令，通知服务器自己处于被动模式。服务器收到命令后，会开放一个大于 1024 的端口 P 进行监听，然后用 PORT P 命令通知客户端，自己的数据端口是 P。客户端收到命令后，会通过 *N*+1 号端口连接服务器的端口 P，然后在两个端口之间进行数据传输。在被动模式下，FTP 的数据连接和控制连接的方向是一致的，也就是说，是客户端向服务器发起一个用于数据传输的连接。客户端的连接端口是发起这个数据连接请求时使用的端口号。

3．FTP 的工作原理

FTP 采用“客户机/服务器（C/S）”方式，客户端需要在自己的本地计算机上安装 FTP 客户软件，服务器端要安装 FTP 服务器方软件。用户通过客户端程序连接到在远程主机上的 FTP 服务器程序。用户通过客户机程序向服务器程序发出命令，服务器程序执行用户所发出的命令，并将执行的结果返回客户机。

例如，用户发出一条命令要求服务器向用户传送一个文件的一份复制，服务器会响应这条命令，将指定文件送到用户的机器上。客户机程序代表用户接收这个文件，将其存放在用户目录中。

FTP 客户端程序有字符界面和图形界面两种，字符界面的命令通常为 ftp，当然也有 lftp、gftp、sftp、ncftp 和 cuteftp 等。gftp 和 cuteftp 是图形界面的，图形界面下最常用的 FTP 客户程序是上网常用的浏览器，通过图形界面可以快捷、方便地进行 FTP 的文件传输等操作。

用户在 FTP 的使用过程中，经常遇“下载”和“上传”的概念。就现在来讲，这两个概念是清楚的，所不同的是下载是常用的，而上传是不经常使用的，通常上传是被禁止的，或只允许个别授权用户使用。以下载文件为例，当用户启动的 FTP 从远程计算机上下载文件时，事实上启动了两个程序：一个是本地机上的 FTP 客户程序（通常为 ftp），它向 FTP 服务器提出下载请求；另一个是在远程计算机上的 FTP 服务器程序（比如 vsftpd），它响应客户端的请求，把指定的文件传送到发出请求的计算机上。

4．匿名 FTP

使用 FTP 时必须首先登录，在远程主机上获得相应的权限后，方可上传或下载文件。换言之，除非有用户名和口令，否则无法传送文件，但这种情况违背了 Internet 的开放性。Internet 上的 FTP 主机数量众多，不可能要求每个用户在每一台主机上都拥有账号。

匿名 FTP 是在 Internet 上发布软件的常用方法。Internet 中有数目众多的匿名 FTP 主机及更多的文件。

匿名 FTP 是这样一种机制，用户可通过它连接到远程主机上，并从其下载文件，而无须成为其注册用户。系统管理员建立了一个特殊的用户——匿名用户，其名为 anonymous 或 ftp，密码为 ftp，网络上的任何人在任何地方都可使用该用户和密码登录到提供匿名服务的 FTP 服务器。

不是所有的 FTP 服务器均支持匿名 ftp。当远程主机提供匿名 FTP 服务时，会指定某些目录向公众开放，允许匿名存取，系统中的其余目录则处于隐匿状态。作为一种安全措施，大多数匿名 FTP 主机都允许用户下载文件，而不允许用户上传文件。即使有些匿名 FTP 主机确实允许用户上传文件，用户也只能将文件上传至某一指定目录中。随后，系统管理员会去检查这些文件，之后会将这些文件移至另一个公共下载目录中，供其他用户下载，避免有人上传有问题的文件。

5．FTP 的典型消息

在 FTP 客户程序与 FTP 服务器进行通信时，经常会看到一些由 FTP 服务器发送的消息，这些消息是 FTP 协议所定义的。表 17-1 给出了一些典型的 FTP 消息。

表 17-1　FTP 协议中定义的典型消息

消　息　号	含　　义	消　息　号	含　　义
125	数据连接打开，传输开始	200	命令 OK
226	数据传输完毕	230	登录成功
331	用户名 OK，需要输入密码	425	不能打开数据连接
426	数据连接被关闭，传输被中断	452	错误写文件
500	语法错误，不可识别的命令	550	文件打开错误

6．FTP 的用户

一般来说，传输文件的用户需要先经过认证后才能登录服务器，然后方能访问、下载或上传文件。根据 FTP 服务器服务的对象不同，可以将 FTP 服务器的使用者分为 3 类：本地用户、虚拟用户（Guest 用户）和匿名用户（ftp）。

（1）本地用户。如果用户在远程 FTP 服务器上拥有账号，则为本地用户。本地用户可以通过输入自己的账号和口令来登录系统。当本地用户登录系统后，其所在位置为用户自己的家目录（$HOME），经配置后本地用户既可以下载又可以上传。

（2）虚拟用户。如果用户在远程 FTP 服务器上拥有账号，但此账号只能用于 ftp 文件传输服务，则称此用户为虚拟用户。虚拟用户可以通过输入自己的账号和口令来进行授权登录。当虚拟用户登录系统后，其所在目录为该虚拟用户的家目录。在通常情况下，虚拟用户既可以下载又可以上传。

（3）匿名用户。FTP 服务器通常设有匿名用户供非本地用户使用。在通常意义上，浏览登录 FTP 网站时使用的是匿名用户，不需要提供账号和密码。但对于字符或命令行界面，就必须提供匿名用户名（ftp 或 anonymous）和口令密码（ftp）。当匿名用户登录系统后，所在的位置为目录/var/ftp（红帽系统）或/srv/ftp（Ubuntu），它是 FTP 服务器上匿名用户的根目录。一般情况下 FTP 服务器对匿名用户只提供下载服务。

17.1.2　Linux 系统的 FTP 服务器

目前，在 Linux 环境下常用的 FTP 服务器有 vsftp、Proftpd 和 Wu-ftpd 等，它们都是基于 GPL。vsftpd 是一个在类 UNIX 系统上使用的 FTP 服务器软件，可以运行在 Linux、BSD、Solaris、HP-UX、IRIX 等系统上。它支持很多其他传统的 FTP 服务器所不支持的特征。vsftpd 为“Very Secrue FTP daemon”的缩写，从此名称可以看出安全性是开发 vsftpd 的初衷，除了与生俱来的安全性能以外，高速、稳定是 vsftpd 的另外两个特性。

在速度方面：使用 ASCII 模式下载数据时，vsftpd 的速度是 Wu-ftpd 的两倍。如果 Linux 的主机使用 2.4 以上版本的内核，在千兆以太网上的下载速度可达 86Mb/s。

在稳定性方面，vsftpd 可以在单机（非集群）上支持 4000 个以上的并发用户同时连接，

它具有以下特点：

（1）是一个安全、高速、稳定的 FTP 服务器。

（2）支持带宽限制。

（3）具有良好的可伸缩性。

（4）支持创建虚拟用户。

（5）支持 Ipng。

（6）可以设置为从 inetd 启动或独立 FTP 服务器两种方式运行。

（7）支持 PAM 或 xinetd/tcp-wrappers 的认证方式。

红帽系列和 Ubuntu 系统均支持 vsftpd，以下我们以 vsftpd 为例说明 FTP 服务器配置及使用。

17.2　vsftpd 服务器

17.2.1　vsftpd 服务器的安装与启动

1．安装 vsftpd

在红帽系列和 Ubuntu 系统中的 vsftp 服务软件包均为 vsftpd，可在安装之前先检测系统是否已经安装了该服务，如果已经安装，则不需要再安装。

检测命令如下：

```
#rpm -q vsftpd                    #红帽系统
#dpkg -l vsftpd                   #Ubuntu 系统
```

若还没有安装，则可以使用以下命令安装：

```
#yum install vsftpd               #红帽系统
#apt install vsftpd               #Ubuntu 系统
```

2．vsftpd 的启动

vsftpd 可以独立（Standalone）方式运行，也可以采用 xinetd 方式运行。这里仅介绍以独立方式运行的配置与管理方法，非独立方式运行情况参见 6.4.3 节超级服务器部分。

vsftpd 服务器的服务名为 vsftpd.service，用户可以使用服务管理的方法管理它的运行。

```
#systemctl status vsftpd                          #检查运行状态
#systmctl enable/disable vsftpd                   #设为开启/关闭开机自启动
#systemctl start/stop/restart/reload vsftpd       #启动/停止/重启动/重载
```

3．测试 vsftpd 服务器

vsftpd 服务器安装并启动服务后，使用默认配置就可以工作了。设 ftp 服务主机名为 ftp_svr，IP 地址为 192.168.16.16。下面使用 ftp 命令登录 vsftpd 服务器 ftpsvr，以检测该服务器能否正常工作。

```
#ftp ftp_svr                        #使用客户端 ftp 命令连接服务器
    Name (ftp_svr:root): ftp        #输入用户名，ftp 为匿名用户名
    Password: ftp                   #输入用户密码，ftp 为匿名用户密码
    ftp>                            #登录成功，出现命令行提示符
    ftp>ls                          #列服务器目录
    ftp>by                          #退出 ftp
```

说明：上述登录未必能成功，因为 Ubuntu 的 vsftpd 服务器默认不支持匿名服务；在红帽系统上也有 SELinux 的控制问题。对于前者可以使用本地用户，对于后者可参见 10.5.8 节。除此之外，还有防火墙问题。

17.2.2 vsftpd 的配置

1．配置文件

在红帽系统与 Ubuntu 系统中，vsftpd 配置文件的位置并不相同，配置文件也有出入。在红帽和 Ubuntu 系统中的配置文件位置或配置目录分别为/etc/vsftpd/和/etc/，还有匿名用户位置分别是/var/ftp/和/srv/vtp/。为了方便描述，我们把配置目录记为 CONF_DIR，也就是说，对于红帽系统 CONF_DIR=/etc/vsftpd/，对于 Ubuntu 系统 CONF_DIR=/etc/。以后在说配置文件时，是基于 VSFTPD_DIR 的，不再强调目录，只说文件。

vsftpd 的常用配置目录及配置文件如表 17-2 所示。

表 17-2　vsftpd 的常见配置目录及配置文件

文件	红帽系统	Ubuntu	说明
vsftpd.conf	vsftpd.conf	vsftpd.conf	主配置文件
ftpusers	ftpusers	ftpusers	
userlist	userlist	vsftpd.user_list	默认不启用
配置目录	/etc/vsftpd	/etc/	记为 CONF_DIR
匿名用户目录	/srv/ftp/	/var/ftp/	记为 anon_root
PAM 配置文件	/etc/pam.d/vsftpd	/etc/pam.d/vsftpd	

vsftpd 服务的主配置文件 vsftpd.conf 是一个文本文件，其中以“#”字符开始的行是注释行，空行为无效行，在分析时将被忽略，其余的行被视为配置命令行。每个选项设置占一行，格式为“option=value”，注意“=”两边不能有空白符。除了这个主配置文件外，还可以给特定用户设定个人配置文件。当 vsftpd 刚刚安装之后，vsftpd.conf 文件中的默认配置是比较简单的，安全性、实用性都不太高，用户可以根据实际情况对其进行一些设置和修改，以使得 vsftpd 应用更具有针对性或目的性。具体做法是直接通过 vi 等文本工具编辑主配置文件，为配置生效，在保存修改后，使用命令 systemctl 重启 vsftpd 的服务，或重载配置。

2．登录和匿名用户的设置

write_enable=YES/NO：是否对登录用户开启写权限，属于全局性设置。

local_enable=YES/NO：是否允许本地用户登录 FTP 服务器。

anonymous_enable=YES/NO：设置是否允许匿名用户登录 FTP 服务器。

ftp_username=ftp：定义匿名用户的账户名称，默认值为 ftp。

no_anon_password=YES/NO：匿名用户登录时是否询问口令，设置为 YES，则不询问。

anon_word_readable_only=YES/NO：匿名用户是否允许下载可阅读的文档，默认为 YES。

anon_upload_enable=YES/NO：是否允许匿名用户上传文件，只有在 write_enable 设置为 YES 时，该配置项才有效。

anon_mkdir_write_enable=YES/NO：是否允许匿名用户创建目录，只有在 write_enable 设置为 YES 时才有效。

3．设置欢迎信息

用户登录 FTP 服务器成功后，服务器可向登录用户输出预设置的欢迎信息，设置如下：

ftpd_banner= Welcome to this FTP service.：该配置项用于设置比较简短的欢迎消息，若欢迎信息较多，则可使用 banner_file 配置项。

banner_file=banner：设置用户登录时，将要显示输出的文件。该设置项将覆盖 ftpd_banner 的设置。

dirmessage_enable=YES/NO：设置是否显示目录消息。若设置为 YES，则当用户进入目录时，将显示该目录中由 message_file 配置项指定的文件（.message）中的内容。

message_file=.message：设置目录消息文件，可将显示信息存入该文件。

4．设置用户登录后所在的目录

local_root=dir：设置本地用户登录后所在的目录。默认配置文件中没有设置该项，此时用户登录后，所在的目录为自己的家目录，对于 root 用户，则为/root 目录。

anon_root=dir：设置匿名用户登录后所在的目录。若未指定，则红帽系统默认为/var/ftp/，Ubuntu 系统默认为/srv/ftp/。

5．控制用户是否允许切换到其他目录

在默认配置下，匿名用户是被限制在根目录/var/ftp（红帽）或/srv/ftp/（Ubuntu）下的，看不到外部世界。但是本地用户却可以通过 cd 命令将工作目录改变到操作系统允许的任何目录，这对系统来说是不安全的，如果这个本地用户是 root，那就更危险了，转化为可以切换到系统中的任何地方并从那里下载文件，或者若设置了 write_enable=YES 允许上传文件的话，还可以上传文件来覆盖本地文件。为了防止这种情况的发生，vsftpd 提供了一种类似操作系统的 chroot 功能，使本地用户登录成功被禁锢在自己的家目录，而自己看到的根目录也只是自己的家目录，即当本地用户不允许切换到其他目录时，成功登录后命令 pwd 的输出为“/”，是该本地账号的家目录。具体设置如下：

chroot_list_enable=YES/NO：设置是否启用 chroot 功能。

chroot_list_file=file：用于指定 chroot 用户列表文件，该文件用于控制哪些用户可以切换到其他目录。默认时，红帽系统为 chroot_list，Ubuntu 系统为 vsftpd.chroot_list。

chroot_local_user=YES/NO：用于指定用户列表文件中的用户是否允许切换到其他目录。

具体情况有以下几种：

（1）当 chroot_list_enable=YES，chroot_local_user=YES 时，在 chroot_list_file 指定的文

件中的用户可以切换到上级目录；没有在文件中的用户，不能切换到站点根目录的上级目录。

（2）当 chroot_list_enable=YES，chroot_local_user=NO 时，在 chroot_list_file 指定文件中的用户不可以切换到其他目录，但其他用户可以切换到其他目录。

（3）当 chroot_list_enable=NO，chroot_local_user=YES 时，所有用户均不能切换到上级目录。

（4）当 chroot_list_enable=NO，chroot_local_user=NO 时，所有用户均可以切换到上级目录。

6．设置访问控制

（1）设置允许或不允许访问的主机。

tcp_wrappers=YES/NO：用来设置 vsftpd 服务器是否与 tcp wrapper 相结合，进行主机的访问控制。默认设置为 YES，vsftpd 服务器会检查/etc/hosts.allow 和/etc/hosts.deny 中的设置，以决定请求连接的主机是否允许访问该 FTP 服务器。比如，若要仅允许 192.168.168.1~192.168.168.254 中的用户访问 vsftpd 服务器，则可在/etc/hosts.allow 文件中添加以下内容：

```
vsftpd:192.168.168. : allow
all: all: deny
```

（2）设置允许或不允许访问的本地用户。

vsftp 还可通过以下配置来控制本地用户的访问，相关的配置如下所述。

userlist_enable=YES/NO：决定由 userlist_file 指定的文件是否启用生效。YES 为生效，NO 为不生效。若为 YES 时，则默认 userlist_file 文件指定文件中的用户将被拒绝。

userlist_deny=YES/NO：决定由 userlist_file 指定的文件中的用户是允许访问还是不允许访问 FTP 服务器。YES 为不允许，NO 为允许。

userlist_file=file，用于指定当 userlist_enable=YES 时使用的文件名，默认时，红帽为 user_list，Ubuntu 为 vsftpd.user_list。

（3）pam.d。

在使用 PAM 编译或构成的 vsftpd 服务器中，还可以使用 PAM 服务来控制本地用户的访问。Linux 的 PAM（Pluggable Authentication Modules，“可插拔式认证模块”），是一种性能健壮而且灵活方便的用户级认证方式，目前，已经成为了 Linux、BSD 和其他一些类 UNIX 操作系统的首选认证方式。特别是在 Linux 上，几乎所有的 daemon 和一些与授权有关的命令都通过 PAM 来进行验证。因此在 vsftpd 服务中，可以使用该服务来对本地用户的访问进行控制。这种控制在 vsftpd.conf 配置文件可通过 pam_service_name 参数来设定。pam_service_name 将指定一个/etc/pam.d/目录中的一个用于控制 vsftpd 服务的文件，一般为 vsftpd。

/etc/pam.d/vsftpd 的内容中有如下一行信息：

```
auth required pam_listfile.so item=user sense=deny file=/etc/vsftpd/ftpusers onerr=succeed    #红帽
auth required pam_listfile.so item=user sense=deny file=/etc/ftpusers onerr=succeed            #Ubuntu
```

这说明 pam_service 将使用 ftpusers 文件，并拒绝其中的用户访问 FTP 服务器。但是，管理员可以修改此行，使用其他文件。

基于此，ftpusers 文件中的用户将被拒绝。

7．设置访问速度

anon_max_rate=0：设置匿名用户所能使用的最大传输速度，单位为字节/s。默认为 0，不受速度限制。

local_max_rate=0：设置本地用户所能使用的最大的传输速度，单位为字节/s。默认为 0，不受速度限制。

8．定义用户配置文件

在 vsftpd 服务器中，不同用户还可以使用不同的配置，这可通过用户配置文件来实现。

user_config_dir=/etc/vsftpd/userconf/：用于设置配置文件所在的目录。

当设置了该配置项后，用户登录 FTP 服务器时，系统就会到 user_config_dir 所指定的目录下读取与当前用户名相同的配置文件，并根据文件中的配置命令，对当前用户进行更进一步的配置。比如，利用用户配置文件，可实现对不同用户访问的速度进行控制等。

9．与连接相关的设置

listen=YES：设置 vsftpd 服务器是否以 standalone 模式运行。设置为 YES 时，以 standalone 模式运行，此为默认值。若设置为 NO，则 vsftpd 可以在 xinetd 服务器的管理控制下运行，功能上会受到限制。

max_clients=0：设置 vsftpd 允许的最大连接数，默认为 0，表示不受限制。若设置为某个非 0 有效值，则超出的请求将拒绝建立连接。只有在 standalone 模式运行时才有效。

max_per_ip=0：设置每个 IP 地址允许与 FTP 服务器同时建立连接的数目，默认为 0，不受限制。通常可对此配置进行设置，防止同一个用户建立太多的连接。只有在以 standalone 模式运行时才有效。

listen_address=IP 地址：设置在指定的 IP 地址上侦听用户的 FTP 请求。若不设置，则对所有可达的 IP 地址进行侦听。对于只绑定了一个 IP 地址的服务器，不需要配置该项。若服务器同时绑定了多个 IP 地址，则应通过该配置项指定在哪个 IP 地址上提供 FTP 服务，即指定 FTP 服务器所使用的 IP 地址。在默认情况下，配置文件中没有该配置项，只有在以 standalone 模式运行时才有效。

accept_timeout=60：PASV 工作方式下设置建立 FTP 链接的超时时间，单位为秒，默认值为 60s。

connect_timeout=120：PORT 方式下建立数据链接的超时时间，单位为秒，默认值为 60s。

data_connection_timeout=120：设置建立 FTP 数据链接后一次数据传输的超时时间，默认为 300s。

idle_session_timeout=600：设置多长时间不对 FTP 服务器进行任何操作，则断开该 FTP 连接，单位为秒，默认为 600s。即设置空闲时间最大值，在这个时间内，既没有命令也没有数据传输。若超过该时间，则会强行断开链接。

pam_service_name=vsftpd：设置在 PAM 所使用的名称，默认值为 vsftpd。在/etc/pam.d/目录内可以看到此文件。

10．FTP 工作方式与端口设置

listen_port=21：设置 FTP 服务器建立连接所侦听的端口，默认值为 21。

connect_from_port_20=YES：默认值为 YES，指定 FTP 数据传输链接使用 20 端口。如果设置为 NO，则进行数据链接时所使用的端口由 ftp_data_port 指定。

ftp_data_port=20：设置 PORT 方式下 FTP 数据链接所使用的端口，默认值为 20。

pasv_max_port=0：设置在 PASV 工作方式下，数据链接可以使用的端口范围的上界，默认值为 0，表示任意端口。

pasv_min_port=0：设置在 PASV 工作方式下，数据链接可以使用端口范围的下界。默认值为 0，表示任意端口。

11．设置传输模式

在默认情况下，vsftpd 是禁止使用 ASCII 传输模式。即使 FTP 客户端使用 ascii 命令，指明要使用 ASCII 模式。下面的选项控制 vsftpd 是否使用 ASCII 传输模式。

ascii_upload_enable=YES/NO：控制是否允许使用 ASCII 模式上传文件，YES 为允许，NO 为不允许，默认为 NO。

ascii_download_enable=YES/NO：控制是否允许使用 ASCII 模式下载文件，YES 为允许，NO 为不允许，默认为 NO。

需要说明的是，当使用 ASCII 方式传输时，可能会因系统间对文本处理的方式不同而导致数据变形。若遇到这种情况，应在客户端执行 binary 命令后，再进行文件传输。

12．设置上传文档的所属关系和权限

（1）设置匿名上传文档的属主。

chown_uploads=YES/NO：用于设置是否改变匿名用户上传文件的属主，默认为 NO。若设置为 YES，则匿名用户上传文件的属主将被设置为 chown_username 配置项所设置的用户名。

chown_username=whoever：设置匿名用户上传文档的属主名。建议不要设置为 root 用户。

（2）新建文件的权限掩码。

local_umask=022：设置本地用户新建文件的 umask，默认为 022，对应的文件权限为 644。022 适合于大多数情况，一般不需要更改。若设置为 077，则对应的权限为 600。

anon_umask=022：设置匿名用户新增文档的 umask。

file_open_mode=755：设置上传文档的权限，权限采用数字格式。

13．日志文件

xferlog_enable=YES/NO：是否启用上传/下载日志记录。

xferlog_file=var/log/vsftpd.log：设置日志文件名及路径。

xferlog_std_format= YES/NO：日志文件是否使用标准的 xferlog 格式。

14．其他设置

text_userdb_names=YES/NO：设置在执行 dir/ls -l 命令时的 UID、GID 显示方式。默认方式为 NO（数字方式），若希望以用户名和组名显示，则设置为 YES。

ls_recurse_enable=YES/NO：是否递归列目录，默认为 NO（不递归），只列当前层目录内容，若需要递归，应将其设置为 YES，则允许执行“ls -R”命令。

17.2.3　vsftpd.conf 的常见应用配置

1．允许匿名用户上传文件

允许匿名用户上传文件，只需在 vsftpd.conf 文件中修改或增加以下选项：

write_enable=YES：匿名用户可写。
anon_world_readable_only=NO：放开匿名用户的浏览权限。
anon_upload_enable=YES：允许匿名用户上传。
anon_mkdir_write_enable=YES：开启匿名用户的写和创建目录的权限。

然后创建供匿名用户上传文件的目录，如 incoming，upload 等，并为其设定权限。在命令行输入下列命令：

```
#cd $anon_root                #进入匿名用户的根目录：红帽为/var/ftp/，Ubuntu 为/srv/ftp
#mkdir incoming               #创建上传目录
#chown ftp:ftp incoming       #修改所有者和组
#chmod g+w incoming           #为同组人增加写权限
```

由于匿名用户（ftp）上传文件时，需要对 incoming 等上传目录拥有写权限，而一般来说，incoming 的所有者为它的创建者（比如 root），建议将它的所有者和组均修改为 ftp，且为同组人增加写权限。

2．配置高安全级别的匿名 FTP 服务器

（1）只允许匿名访问，不允许本地用户访问。

```
anonymous_enable=YES          #激活匿名用户
local_enable=NO               #禁止本地用户登录
```

（2）使用 ftp_banner 取代 vsftpd 默认的欢迎词，免得泄露 FTP 服务器相关信息。

```
ftpd_banner=welcome to my FTP server
```

（3）只让匿名用户浏览可阅读的文件，不可以浏览整个系统。

```
anon_world_readable_only=YES
```

（4）隐藏文件的所有者和组信息，使匿名用户看到文件的所有者和组全变为 ftp。

```
hide_ids=YES
```

（5）取消写权限。

```
write_enable=NO
anon_upload_enable=NO
anon_mkdir_write_enable=NO
anon_other_write_enable=NO
```

（6）使用独立运行模式，并指定监听的 IP 地址或端口号。

```
#listen=YES                       #独立运行模式
#listen_address=ip address        #指定 IP 地址。若本行不存在或被注释则采用默认
#listen_port=PORT                 #指定端口号。若本行不存在或被注释则采用默认 21
```

说明：在 systemd 的控制下，只要 vsftpd.service 工作正常，不需要 listen=YES。

（7）对连接进行控制，还有超时时间，需根据具体情况确定。

```
connect_from_port_20=YES
accept_timeout=60
connect_timeout=120
data_connection_timeout=120
```

（8）控制并发数，限定每个 IP 地址的并发数，根据用户需求设定。

```
max_clients=numerical_value
max_per_ip=numerical_value
```

（9）限定下载速度，具体限多少，可以根据需要确定。

```
anon_max_rate=80000
```

（10）启用详细的日志。

```
xferlog_enable=YES
```

3．配置虚拟用户

允许本地用户上传文件是不安全的，且本地用户所使用的工作目录和权限有很大的不同，不便用于 FTP 服务器数据的上传，而匿名用户的任务是用来下载的。因此还必须设置一些用户用于向 FTP 服务器上传文件供大家下载，vsftpd 提供了虚拟用户功能。ftp 虚拟用户的配置过程如下：

（1）建立虚拟用户及口令库文本文件。

```
#cd $CONF_DIR            #进入配置目录：红帽为/etc/vsftpd/，Ubuntu 为/etc/
#vi vlogin.txt           #编辑一个文本文件，用于存放虚拟用户的名字和密码
```

该文件的奇数行是虚拟用户名，偶数行是密码，每个虚拟用户占 2 行，其中可以包含多个用户信息。比如，虚拟用户名为 virftp01、密码为 virtualftp，此文件内容为：

```
virftp01
virtualftp
```

（2）生成 PAM 认证使用的数据库文件。

此处需要 db_load 命令的支持，若系统中无此命令，则需要安装它。在红帽和 Ubuntu 系

统中 db_load 分别归属为 libdb-utils 和 db-util 软件包。安装方法为：

```
#yum install libdb-utils                    #红帽系统
#apt install db-util                        #Ubuntu 系统
```

安装完毕之后，可以使用命令：

```
#db_load -T -t hash -f vlogin.txt vlogin.db
```

（3）修改权限。

```
#chmod 600 vlogin.db
```

（4）建立虚拟用户所需的 PAM 配置文件。

编辑/etc/pam.d/vsftpd，注释掉其中的所有内容，然后添加以下 2 行：

```
auth required pam_userdb.so db=$CONF_DIR/vlogin
account required pam_userdb.so db=$CONF_DIR/vlogin
```

提醒：别忘了将$CONF_DIR 替换为它的值。红帽系统为/etc/vsftpd/，Ubuntu 为/etc/。

（5）建立虚拟用户所使用的系统用户并为其设置权限。

```
#useradd -M -d $anon-root -G ftp -s /usr/sbin/nologin virftp    #创建宿主用户
#cd $anon-root
#mkdir virftp01                                                 #虚拟用户工作目录
#chown -R ftp:ftp virftp01                                      #修改 vsftpd 工作目录的主组
#chmod -R g+w virftp01                                          #虚拟用户工作目录
```

（6）在主配置 vsftpd.conf 尾部添加如下内容：

```
guest_enable=YES                                #启用虚拟用户
guest_username=virftp                           #虚拟用户名为 virftp
pam_service_name=vsftpd                         #pam 服务名（若已指定可不再做）
user_config_dir=$CONF_DIR/virusers_conf         #虚拟用户的子配置文件目录
```

（7）创建子配置文件目录并生成虚拟用户使用的子配置文件。

```
#mkdir $CONF_DIR/virusers_conf; cd $CONF_DIR/virusers_conf
#编辑文件 virftp01，在其中输入以下内容
anon_world_readable_only=NO                     #用户可以浏览目录和下载文件
anon_other_write_enable=YES                     #用户具有文件改名和删除文件的权限
anon_mkdir_write_enable=YES                     #用户具有目录的创建和删除权限
anon_upload_enable=YES                          #可以上传文件
local_root=$anon_root/virftp01                  #设置虚拟用户工作目录
```

（8）配置修改完成后，重启 vsftpd 服务。

```
#systemctl restart vsftpd
                              或
#systemctl reload vsftpd
```

之后就可以使用虚拟用户登录了。虚拟用户的权限取决于配置文件的设置。本例就配置了一个可创建和删除目录、上传和下载文件的权力很大的虚拟用户 virftp01。

说明：可以配置多个虚拟用户，每个虚拟用户可以在$anon_root 指定的目录下有自己的

配置子文件以配置不同权限；当启用虚拟用户后，本地用户也被归类为虚拟用户，因此当本地用户登录成功后，也工作在（guest_username 的值指定的）虚拟用户的目录内，为了安全可将虚拟用户目录的权限设置小一些，比如 700，即使本地或其他用户登录到虚拟用户目录，不能从其中看到或取走东西；虚拟用户的密码被以文本方式保存在 vlogin.txt 文件中，很容易被泄露，因此，在使用 db_load 生成数据库后应立即删除 vlogin.txt 文件。

17.3　FTP 服务器的使用

vsftpd 服务器配置好后，用户就可以访问该服务器了。设服务器的机名为 ftp_svr，IP 地址为 192.168.16.16，访问方法简介如下。

17.3.1　用浏览器访问

（1）匿名用户登录方式：在浏览器地址栏中输入 FTP 服务器 IP 地址或主机名。

```
ftp://192.168.16.16                 #地址据实际情况而定
ftp://ftp_svr                       #服务器名根据实际情况而定
```

（2）用户登录方式：在浏览器地址栏中输入 FTP 服务器 IP 地址或主机名，且在主机或 IP 地址前指定本地用户名。

```
ftp://root@192.168.16.16            #以 root 用户为例
ftp://root@ftp_svr
ftp://test@ftp_svr                  #以 test 用户为例
```

之后，提示用户输入密码，也可能根据浏览器的不同，首先提示用户再次输入用户名，一切正确之后，才可能登录到 ftp 服务器。当受 ftpusers、user_list、防火墙等限制时，还可能登录不成功，或成功后也无法正常工作。

17.3.2　使用客户端命令 ftp

1．FTP 客户端命令 ftp

使用 FTP 客户端软件访问 FTP 服务器也有两种方式：匿名用户登录方式和本地用户登录方式。Windows/Linux/UNIX 系统都有自己的 ftp 客户端命令，且在用法上有略微的差异。使用前要先安装 ftp 客户端，安装方法如下：

```
#yum install ftp                    #红帽系统
#apt install ftp                    #Ubuntu 系统
```

ftp 命令的通常用法为：

```
ftp [参数] ftp_svr
ftp [参数] ftp_IP
```

Linux ftp 命令的常用参数如表 17-3 所示，常用命令如表 17-4 所示。

表 17-3　ftp 命令的常用参数

参　　数	功 能 描 述	参　　数	功 能 描 述
ftp_srv	FTP 服务器的主机名	ftp_IP	FTP 服务器的 IP 地址
-d	使用调试方式	-g	取消全局文件名
-n	限制 ftp 的自动登录	-i	取消多文件传输时的交互操作

表 17-4　ftp 命令的常用内部命令

命　　令	功 能 描 述
! [cmd [args]]	执行交互 shell[命令 cmd]
?/help [cmd]	显示 ftp 内部命令[cmd]的帮助信息
append l_file [r_file]	将本地文件 l_file 追加到远程文件 r_file 或 l_file
ascii	使用 ASCII 类型传输方式
binary	使用二进制文件传输方式
bye/quit	退出 ftp
case	在使用 mget 时，将远程主机文件名中的大写字母转化为小写字母
cd r_dir	进入远程主机目录 r_dir
cdup	进入远程主机目录的父目录，相当于 cd ..
chmod mode FILE	将远程主机文件 FILE 的存取方式设置为 mode
close/disconnect	关闭与远程服务器的链接会话（与 open 对应）
cr	使用 ASCII 方式传输文本文件时，将回车+换行转换为换行
delete r_file	删除远程主机文件
debug [value]	设置调试方式。value 为 0 时表示取消 debug
dir [r_dir] [l_file]	列出服务器目录[r_dir]，并将结果存入本地文件[l_file]
get/recv r_file [l_file]	将文件 r_file 下载至本地[保存为 l_file]
lcd [dir]	将本地工作目录切换至 dir
ls [r_dir] [l_file]	列出远程目录[r_dir]，并存入本地文件[l_file]
mdelete r_file	删除远程主机文件 r_file
mdir r_files l_file	与 dir 类似，但可指定多个远程文件
mget r_files	下载多个远程文件
mkdir dir_name	在远程主机中建一个目录
mls r_file l_file	同 nlist，但可指定多个文件名
mode [MODE]	将文件传输方式设置为 MODE，默认为 stream 方式
modtime FILE	显示远程主机文件 FILE 的最后修改时间
mput l_file	上传多个文件至远程主机
newer FILE	如果远程主机中的 FILE 比本地同名文件更新，则重传该文件
nlist[r_dir][l_file]	列出远程目录[r_dir]清单并存入本地文件 l_file
open host/IP [port]	建立到指定 ftp 服务器 host/IP 的连接[，可指定端口为 port
passive	进入被动传输方式

（续表）

命　　令	功 能 描 述
prompt	打开或新装多个文件传输时的交互提示
put/send l_file [r_file]	将本地文件 l_file 传送至远程主机[并更名为 r_file]
pwd	显示远程主机的当前工作目录
reget r_file[l_file]	类似于 get，但若 l_file 存在，则从上次传输中断处续传
reset	清除回答队列
restart marker	从指定位置 marker 处重新开始 get 或 put，如 restart 300
rmdir dir_name	删除远程主机目录 dir_name
size FILE	显示远程主机文件 FILE 的大小
Status	显示当前 ftp 状态
system	显示远程主机的操作系统类型
type [TYPE]	设置文件传输类型为 TYPE 或显示当前传输类型
umask [MASK]	将远程服务器的 umask 设置为 MASK 或显示当前的 umask
user user_name [password]	向远程主机表明自己的身份，需要口令时，必须输入口令

2．ftp 命令使用示例

1）交互式

以下示例交互方式，以匿名用户（用户名为 anonymous 或 ftp，密码为 ftp）登录，登录名为 ftp_svr 或 IP 地址为 192.168.16.16 的交互过程。在此过程中，使用了列目录命令 ls，文件上传命令 put test1.txt 和文件下载命令 get test2.txt，最后使用 by 命令退出 ftp 客户端。

```
#ftp 192.168.16.16                          #或  ftp ftp_svr
     Name (192.168.168.154:root): test      #可为匿名用户(ftp)或本地用户
     Password:                              #输入用户密码，匿名用户的密码为 ftp
  ftp> ls                                   #列目录
  ftp> put test1.txt                        #上传 test1.txt 到 ftp 服务器。匿名用户一般不允许上传
  ftp> put test1.txt test2.txt              #上传 test1.txt 到 ftp 服务器，保存为 test2.txt
  ftp> get test2.txt                        #从 ftp 服务器中下载 test2.txt 文件
  ftp>by                                    #退出 ftp 客户程序
```

匿名用户登录时，使用 ftp 或 anonymous 作为 ftp 登录的用户名，ftp 为密码。默认匿名用户不能上传文件。

以上过程也适用于本地用户登录。如果使用本地用户 test 登录系统，则在以上交互过程中，只需要将用户名和密码换成 test 及其密码就可以了。在默认情况下，本地用户登录成功后，所使用的目录是自己的家目录。

需要注意的是，使用本地用户登录存在一定的隐患。首先它是系统账号，一旦 FTP 服务器出现安全漏洞会对整个 FTP 服务器所在的主机系统造成威胁；其次，若 FTP 服务器配置不当，本地用户账号可以离开自己的家目录，到系统中的其他目录去，还可能会允许向从自己家目录以外的目录传送文件。这些都是对系统的安全性威胁，尤其当此本地用户为权力较大的用户时，更是如此。如果该本地用户为 root，那就更危险了，这也是很多系统中拒绝 root 以本地用户身份登录 ftp 的原因。为此，应该设置 FTP，将本地用户禁锢在自己的家目录中，

不让其外出。

2）在 shell 脚本中使用 ftp 客户端程序

在交互方式下，由于有人的参与，ftp 的效率比较低，可以将上述交互过程中使用的命令放在一个脚本文件中，然后在 shell 命令下以 I/O 重定向办法实现整个过程的自动控制。以上进入 ftp 命令后的交互过程命令（去除注释）为：

```
user test testtest
ls
put test1.txt
put test1.txt test2.txt
get test2.txt
by                              #退出 ftp 客户端程序，作为脚本的此句可以不要
```

若将这些脚本保存到 ftp.scrpt1 文件中，则使用以下方式：

```
#ftp -n 192.168.16.16 < ftp.scrpt1              或
#ftp -n ftp_svr < ftp.scrpt1
```

执行之。也可以使用管道，比如：

```
#cat ftp.script1 | fpt -n ftp_svr
```

在 ftp.script1 中使用了 user 命令，它所带的两个参数分别为用户名和用户密码。在 ftp 的命令行中还使用了-n 参数，以禁止 ftp 的自动登录功能，因为在脚本中还要指定具体的用户和密码。若要在多文件传输与操作中（mput/mget/mdelete）禁止交互与提示，还可以使用-i 选项，或在脚本中使用 prompt 命令。

上述 ftp 命令行中的主机或 IP 信息也可以添加到 ftp 的脚本文件，此时可在脚本文件开始处添加一行打开通信链路命令：

```
open ftp_svr                    或
open 192.168.16.16
```

3）使用即时文档执行 ftp 客户端程序

将 ftp.script1 的内容作为即时文档，也可以执行 ftp 客户端脚本命令。下面提供一个新的任务：使用 test 用户（密码为 tsettset），从 ftp_svr 上下载所有的*.txt 文件。一种实现方法如下：

```
#ftp -n -i << !
    open ftp_svr
    user test tsettset
    prompt
    mget *.txt
!
```

不论是使用脚本文件或即时文档，在其中使用 user 命令时，总要在文本中写出用户名和密码，这是非常不安全的，应尽量避免。

17.4　TFTP 与 TFTP 服务器的使用简介

17.4.1　TFTP 协议

TFTP（Trivial File Transfer Protocol，简单文件传输协议）是 TCP/IP 协议族中的一个用来在客户机和服务器之间进行简单文件传输的协议。

TFTP 通常基于 UDP 协议实现，是为了进行小文件传输，因此它不具备一般 FTP 的许多功能，因此，可以看成是一个简化了的 FTP。它与 FTP 的主要区别是没有用户权限认证与权限管理等功能。也就是说 TFTP 不需要验证客户端的身份及权限。因此 TFTP 的缺点是传送不可靠、没有用户和密码验证等，尽管如此，它仍非常适合安全性要求不高的小型文件传送。

TFTP 支持文件的上传与下载功能。如果客户端发送的是下载请求，那么，服务器将根据客户端发过来的报文，解析出文件的路径和文件名，并且根据解析出来的文件名及内容开始构造报文，然后把它发送给客户端；如果客户端发送来的是上传请求，那么，服务器端也必须解析出文件名及要保存的路径。若满足条件要求，则发送 ACK 给客户端以确认已经接受客户端的请求，然后等待客户的报文。客户端接收 ACK 后，就可以开始发送数据报文，服务器开始接收报文，并将其写到指定路径的文件中。

在 TFTP 中，任何一个读取或写入文件的请求也是连接请求。如果服务器批准此请求，则服务器打开连接，数据以定长 512 字节传输。每个数据包都包括一块数据，服务器发出下一个数据包之前，数据块必须得到客户对上一个数据包的确认。如果一个数据包的大小小于 512 字节，则表示传输结束。如果数据包在传输过程中丢失，发出方会在超时后重新传输最后一个未被确认的数据包。通信的双方都是数据的发出者和接收者，一方传输数据接收应答，另一方发出应答接收数据。

在 TFTP 文件的传输过程中，通常都要求有一定的容错能力。大部分的错误都会导致连接中断，假定错误由一个错误的数据包引起，则这个包不会被确认，也不会被重新发送，因此，另一方无法接收到。如果错误包丢失，则使用超时机制。一般的错误主要是由下面三种情况引起的：一是不能满足请求；二是收到的数据包内容错误，而这种错误又不能由延时或重发解决；三是对需要的资源访问错误（如磁盘满等）。

17.4.2　TFTP 的安装和配置及应用

1．TFTP 的安装

TFTP 服务的客户端程序 tftp 归属软件包为 tftp，而服务器软件包在红帽和 Ubuntu 下分别为 tftp-server 和 tftpd，可以使用以下命令安装：

```
#yum install tftp tftp-server                    #红帽系统
#apt install tftp tftp-server                    #Ubuntu 系统
```

2．TFTP 服务器的设置

1）红帽系统

在红帽系统中，tftp 服务可以独立运行，其服务名为 tftp.service，服务器端根目录为/var/lib/tftpboot。服务管理方法为：

```
#systemctl enable tftp.service                        #启用 tftp.service
#systemctl start/stop/restart tftp.service            #启动/停止/重启
#systemctl status tftp.service                        #检查状态
```

2）Ubuntu 系统

Ubuntu 系统提供的服务器 tftpd 软件包没有提供由 systemctl 管理的组件，只能按传统方法交由超级服务器 xinetd 来管理。

首先安装 xinetd 软件包，并启动 xinetd 服务，方法如下：

```
#yum install xinetd
#systemctl enable xinetd.service                      #启用 tftp.service
#systemctl start/stop/restart/reload xinetd.service   #启动/停止/重启/重新加载配置
```

其次，进入/etc/xinetd.d，在其中创建文件 tftp，使其内容为：

```
service tftp
{
socket_type         = dgram
protocol            = udp
wait                = yes
user                = root
server              = /usr/sbin/in.tftpd
server_args         = -s /var/lib/tftpboot
disable             = no
per_source          = 11
cps                 = 100 2
flags               = IPv4
}
```

最后，重启或重载 xinetd 服务器，方法如下：

```
#systemctl reload xinetd.service     #重载或
#systemctl restart xinetd.service    #重启
```

说明：此方法也可用于红帽系统，方法相同，但不能启用 tftp.service。

3．tftp 命令

tftp 命令的用法为：

```
tftp [options] [host [port]] [-c cmd]                 #红帽系统
tftp [host]                                           #Ubuntu 系统
```

参数 host 用于指定主机名或主机的 IP 地址，-m mode 用于指定传输方式；-c cmd 用于指

定子命令。tftp 的常用子命令如表 17-5 所示。

表 17-5 tftp 的常用命令

命　　令	功 能 描 述	命　　令	功 能 描 述
connect host [port]	连接到远程 tftp 服务器 host	status	显示当前状态信息
put l_file [r_file]	上传文件 l_file 为 r_file	get r_file [l_file]	下载文件 r_file 为 l_file
binary/ascii	设置二进制/文本传输模式	mode ascii/ binary	ascii/ binary 传输模式
rexmt scnds	设置包传输的超时时限为 scnds	timeout seconds	设置总传输超时时限为 seconds
trace	调试模式（跟踪包路径，大小等）	verbose	设置详细模式
quit	退出 tftp	help/? cmd	对命令 cmd 进行帮助

4．TFTP 服务器的应用

1）用于远程系统引导或远程备份

可以利用 TFTP 与 DHCP、FTP 服务器配合，构成为一组 Linux 安装服务器或备份服务器。利用安装服务器可以进行大规模的 Linux 操作系统网络安装。在网环境中，还可以利用 Linux 的 TFTP 服务器下载路由器配置文件，编辑后再上传到路由器。

2）利用 TFTP 进行文件的传输

假定 tftp 服务器的机名为 tftp_srv，IP 地址为 192.168.16.16，要下载和上传的文件分别为 tftp_down 和 tftp_up。由于上传和下载的用法是相同的，所不同的是所使用的命令为 get 或 put，这里仅以下载为例说明之。

（1）命令行方式

```
#tftp 192.168.16.16 -c get tftp_down            #仅红帽系统
#tftp tftp_srv -c get tftp_down
```

（2）交互方式

```
#tftp 192.168.16.16                             #登录成功会显示 tftp 提示符，进入交互界面
 tftp>get tftp_down                             #下载文件 tftp_down
 tftp>quit                                      #退出 tftp 客户端
```

（3）即时文档方式

```
#tftp 192.168.16.16 << EOF
  get tftp_down
  quit
EOF
```

17.5 与防火墙和 SELinux 的关系

17.5.1 与防火墙的关系

需要设置防火墙规则，让其允许 ftp 和 tftfp 通过。

1．firewalld

```
#firewall-cmd --add-service=ftp          #允许 ftp
#firewall-cmd --add-service=tftp         #允许 tftp
```

2．ufw

```
#ufw allow ftp                           #允许 ftp
#ufw allow tftp                          #允许 tftp
```

17.5.2　与 SELinux 的关系

SELinux 与 ftp 的关系见于 10.5.8 节，这里只说一下 SELinux 与 tftp 的关系。

首先，使用命令

```
#getsebool -a | grep tftp
```

得到 tftp 与 SELinux 相关的布尔变量有：

```
tftp_anon_write --> off
tftp_home_dir --> off
```

它们都是没有开启的，若要具有上传功能，除了使其“根”目录/var/lib/tftpboot 具有可写权限外，还要打开 tftp_home_dir 布尔变量，方法是：

```
#chmod a+w /var/lb/tftpboot
#setsebool -P tftp_home_dir 1
```

习题 17

1．单项选择题

（1）默认情况下，匿名 ftp 站点的主目录是（　　）或（　　）。

A．/run/ftp　　B．/var/ftp　　C．/srv/ftp　　D．/etc/ftp

（2）在 TCP/IP 模型中，应用层包含了所有的高层协议，在下列一些应用协议中，能够实现本地与远程主机之间文件传输工作的协议是（　　）。

A．TELNET　　B．FTP　　C．SNMP　　D．NFS

（3）重启 vsftpd 服务的命令是（　　）。

A．systemctl enable vsftpd　　B．systemctl start vsftpd

C．systemctl restart vsftpd　　D．systemctl status vsftpd

（4）以下不属于 FTP 客户端命令的是（　　）。

A．date　　B．get　　C．put　　D．bye

2．简述题

（1）简述 FTP 服务的工作原理。

（2）简述 ftpusers 文件的位置及作用。
（3）简述 TFTP 服务的功能。

实验 17

1．通过以下方式访问 FTP 服务器访问：
（1）浏览器匿名访问、本地用户访问；
（2）客户端匿名访问、本地用户访问（交互式）；
（3）客户端匿名访问、本地用户访问（脚本或即时文档）。
2．利用 vsftpd，配置一台 FTP 服务器，要求实现以下功能：
（1）本地用户可以上传、下载，且可改工作目录到任何地方；
（2）匿名用户只能下载。
3．利用 vsftpd，配置一台 FTP 服务器，要求分别实现：
（1）建立宿主用户 virftp；
（2）实现虚拟用户 vftp1，vftp2 上传、下载和删除，虚拟用户所使用的目录自行指定。

第 18 章　网络资源共享服务

18.1　网络资源共享简介

随着网络技术和相关应用的飞速发展，网络上的信息资源呈爆炸型趋势，日益增长的信息资源给人类社会带来了极大的方便，使人们在不同硬件平台、不同的操作系统平台间可以进行交流、通信、共享信息。这些不同硬件或不同操作系统平台称为异质环境。这些异质系统在数据存储的格式、文件的组织形式、访问权限等方面都存在很大的差异，这使得它们之间的数据共享存在一定的问题。网络存储服务器必须能为这些不同操作系统平台上的客户端提供远程文件访问和数据共享功能，使得只要权限允许，不管是何种平台上的用户都可以方便地共享信息资源。

网络资源共享是指服务器上的驱动器、文件、数据资源等能够被网络上的用户访问，使计算机网络中的用户互通有无，分工协作，从而大大提高硬件和数据的利用率。

第 17 章所介绍的 ftp 和 tftp 是网络资源共享的常见形式。目前，被广泛应用于异质环境中的信息共享协议还有 SMB（Server Message Block，服务信息块）和 NFS（Network File System，网络文件系统）。前者常用于 Windows 与 OS/2 的网络共享，后者则常用于 UNIX/Linux 间的共享。有了 Samba，可使 Windows 与 UNIX/Linux 间的共享变得非常轻松、简便。

18.2　Samba服务

18.2.1　SMB 协议与Samba

1．SMB 协议

SMB 协议是实现网络上不同类型计算机之间文件和打印机共享服务的协议。

在 NetBIOS 出现以后，Microsoft 就是使用 NetBIOS 实现了一个网络文件/打印服务共享系统，这个系统基于 NetBIOS 设定了一套文件共享协议，即 SMB 协议，这个协议被 Microsoft 用于 Lan Manager 和 Windows NT 服务器系统中。Windows 系统均包括这个协议的软件，因此，这个协议在局域网系统中影响很大。

随着 Internet 的流行，Microsoft 希望将这个协议扩展到 Internet 上，成为 Internet 上计算机之间相互共享数据的一种标准。它将原有的几乎没有多少技术文档的 SMB 协议进行整理，重新命名为 CIFS（Common Internet File System，通用 Internet 文件系统），并打算将它与 NetBIOS 相脱离，试图使它成为 Internet 上的一个标准协议。

为了让 Windows 和 UNIX 实现资源共享，最简单的办法是在 UNIX 中安装支持 SMB/CIFS 协议的软件，这样 Windows 客户就不需要更改设置，就能如同使用 Windows 服务器一样访问 UNIX/Linux 系统资源。所以，SMB/CIFS 协议提供了在网络上不同计算机系统之间共享文件和打印机的方法。

SMB 使用 NetBIOS API 实现面向连接的协议，该协议为 Windows 客户程序和服务提供一个通过虚电路按照请求/响应方式进行通信的机制。SMB 的工作原理就是让 NetBIOS 与 SMB 协议运行在 TCP/IP 上，并且使用 NetBIOS 的名字解释器，让 Linux 机器可以在 Windows 的“网上邻居”中被看到，同时 Linux 也可看到 Windows“网上邻居”中的主机。通过 SMB 协议，客户端应用程序可以在各种网络环境下读写服务器上的文件，以及对服务器提出服务请求。此外，通过 SMB 协议，应用程序还可以访问远程服务器端的文件和打印机等资源。图 18-1 给出了一个简单使用 SMB 协议的网络结构图。

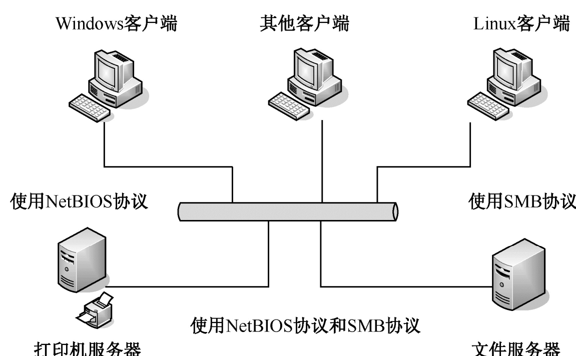


图 18-1　一个简单的使用 Samba 服务器的网络结构图

2．Samba 简介

Samba 是一组软件包，使 Linux 支持 SMB 协议，目前已经成为各种 Linux 发行版本中的一个基本的软件包，几乎可以在所有的类 UNIX 平台上运行，当然包括 Linux。Samba 基于 GPL 发行，当前稳定版本是 4.6.3，最新信息和产品可以从 https://www.samba.org/ 得到。

Samba 协议是在 TCP/IP 上实现的，它是 Windows 网络文件和打印共享的基础，负责处理和使用远程文件和资源。在默认情况下，Windows 工作站上的 Client 使用服务消息块（SMB）协议。正是由于 Samba 的存在，使得 Windows 和 Linux 可以沟通并互相通信。安装了 Samba 后，就可以直接而方便地在 Linux 和 Windows 之间共享资源。

Samba 的核心是 smbd 和 nmbd 两个守护进程。smbd 监听 139 TCP 端口，nmbd 监听 137 和 138 UDP 端口。smbd 是 Samba 的主启动服务器，主要提供文件系统和打印机的共享。mbd 提供解析功能，主要作用是对外发布 Samba 服务器可以提供的服务。

Samba 的主要功能如下：

（1）提供 Windows 操作系统风格的文件和打印机共享。Windows 系统以此共享 Linux 和其他操作系统的资源，外表看起来和共享 Windows 资源没有区别，并支持 Windows 客户使用网上邻居浏览资源。

（2）提供 SMB 客户功能。利用 Samba 提供的 smbclint 程序，可以在 Linux 系统下以类似于 ftp 的方式访问 Windows/Linux 的资源。

（3）备份 PC 上的资源。利用一个叫 smbtar 的 shell 脚本，可以使用 tar 格式备份和恢复远程 Windows 上的共享文件。

（4）提供一个命令行工具。在其上可以有限制地支持 Windows 的某些管理功能。

18.2.2　Samba 的安装与启动管理

1．软件安装

与 samba 相关的软件包如下所述。

服务器：samba
CIFS 工具包：cifs-utils
客户端：samba-client（红帽系统）/smbclient（Ubuntu 系统）
图形界面配置工具：system-config-samba（CentOS/RHEL 无）
访问 Windows 所需软件包：samba-winbind（红帽系统）/ winbind（Ubuntu 系统）

其中的 system-config-samba 不是必需的，可以使用以下命令查询安装情况：

```
#rpm -qa | grep -E '(samba)|(cifs)'                    #红帽系统
#dpkg -l | grep -E '(samba)|(cifs)'                    #Ubuntu
```

若还没有安装，用户可以根据需要安装它们的全部或部分，安装方法为：

```
#yum install samba-client samba samba-winbind system-config-samba cifs-utils     #红帽系统
#apt install smbclient samba winbind system-config-samba cifs-utils              #Ubuntu
```

其他相关的软件包会被自动安装，但有时安装程序还会建议安装一些其他包。

2．服务管理

在红帽和 Ubuntu 下，Samba 的服务分别为 smb.service 和 samba.service，管理方法如下：
1）红帽系统

```
#systemctl status smb.service                    #检查运行状态
#systemctl enable/disable smb                    #启用/禁用
#systemctl start/stop/restart/reload smb         #启动/停止/重启动/重载
```

2）Ubuntu 系统

```
#systemctl status samba.service                  #检查运行状态
#systemctl enable/disable samba                  #启用/禁用
#systemctl start/stop/restart/reload samba       #启动/停止/重启动/重载
```

说明：
尽管 Ubuntu 提供了 samba.service，甚至还有一个 samba-ad-dc.service，但它们并不能很好地工作。若在使用以上方法遇到问题时，可用/etc/init.d/samba 管理 samba 服务，方法如下：

```
#/etc/init.d/samba restart/start/stop/reload     # 重启/启动/停止/重载配置
```

例如，重启 samba 服务执行命令：

```
#/etc/init.d/samba restart                       # 重启 samba 服务
```

18.2.3　Samba 的配置

1．Samba 的配置文件

Samba 服务器的配置文件为/etc/samba/smb.conf。在 Samba 安装后，就有一个可以默认使用的样本程序，在 Ubuntu 下还有较详细的说明。文件中的空行、以“#”或“;”开始的行为注释行，用户可通过阅读该配置文件的内容或通过命令“man smb.conf”，了解其中的配置项及其意义。

配置文件的格式，类似 Windows 系统的*.ini 文件，以节为单位进行配置。该文件被分隔成若干节，每一节都由一个被方括号括起来的标识开始（比如[global]、[homes]和[printers]等）。[global]用于全局参数设置，以控制 Samba 的特性或为其他节提供默认参数。除[global]节外，每个节描述一个共享资源，节的名字就是共享资源的名字，共享资源的属性由节内的参数或选项来描述。Samba 配置文件的常用节如表 18-1 所示。

表 18-1　smb.conf 的节

名　称	说　明
[global]	基本节，用于定义全局变量或其他节变量的默认值
[homes]	用户家目录节，用于定义每个用户家目录的共享方式
[printers]	共享打印机节，用于定义共享打印机
[print$]	用于说明打印机驱动程序的存放位置
[sharename]	共享节。用于自定义共享资源，sharename 由用户指定，可有多个

smb.conf 文件中的每一有效行可以是节的名字或形如“参数=值”的对。每个节都以一个包含在方括号中的节名开始，节名后是“参数=值”对，一行一对，用于定义或描述节的属性和作用。节名和参数名都不区分大小写，但是字符串参数值仍要区分大小写。参数值可以是布尔值或字符串，在含有空格符的字符串也不必用引号说明，如果在进行语法分析时发现引号，将忽略；当参数值含有多项时，用空格符作为字符串之间的分隔符；当参数为布尔型时，它的值可以是 yes/no 或 true/false。

在参数的值中还可以使用变量。Samba 提供了一组固定的变量，都以“%”作为前缀，在执行过程中用其值代替。smb.conf 常用变量如表 18-2 所示。

表 18-2　smb.conf 文件常用变量

变　量	说　明	变　量	说　明
%u	当前服务的用户名	%U	当前对话的用户名
%g	%u 表示的用户所在组的名字	%G	%U 表示的用户所在组的名字
%S	当前服务名	%v	Samba 版本
%H	%u 的宿主目录	%h	Samba 服务器的 internet 主机名
%i	本地被客户端连接的 IP 地址	%I	客户机的 IP 地址
%m	客户机的 NetBIOS 名	%M	客户端 internet 主机名
%P	当前服务的根目录	%T	当前日期和时间
%d	当前服务进程的 PID	%L	服务器的 NetBIOS 名

2．Samba 的配置

1）基本节

基本节[global]用于定义全局变量或为其他节变量提供默认值，常见内容如下所述。

（1）基本选项。

workgroup=WORKGROUP：工作组名称。红帽默认 SAMBA,Ubuntu 默认 WORKGROUP。建议设置相同。

server string = Samba Server Version %v：服务器描述字符串：可以是版本号%v 或主机名%h。可不设置。

passdb backend = tdbsam：指定密码文件。可取值有 tdbsam 或 ldapsam，默认为 tdbsam。此时密码文件为/var/lib/samba/private/passdb.tdb。

interfaces = lo eth0 192.168.12.2/24 192.168.13.2/24：网络接口。可设置多个网络接口和子网或不设。

（2）日志选项。

log level=LEVEL：日志等级。在记录 Samba 活动日志时，设置调试级别。默认值为 0，提高级别可以记录更多信息。

log file = /var/log/samba/log.%m：日志文件位置及名称。

max log size = 50：日志文件最大长度。超过者将被回滚。

（3）安全选项。

security=type：设置 Samba 的安全类型。安全类型可设置为以下几种。

① user（用户）：此为最常见的安全类型，也是 Samba 的默认设置。若要使该模式能够正常工作，最简单的办法就是为每个使用 Samba 的用户设置一个本地用户。在登录服务器之前，需要提供一个用户名和密码确认。于是，通过 Samba 提供给一个用户账户的文件许可与该用户直接登录到 Linux 上时所得到的许可相同。

② share（共享）：此模式最适合打印共享或提供更为公开的文件访问情况。客户端不需要提供有效用户名和密码就可访问服务器。然而，若要访问或修改文件，通常要求文件对用户具有“guest”的读/写许可权。

③ server（服务器）：从客户端的观点看，这个选项与 user 安全模式相同，因为客户端始终必须提供一个有效的用户名/密码组合才能使用 Samba 服务器，区别是在服务器端。使用该安全模式，用户名/密码被发送到另一个有效的 SMB 服务器进行验证，如果验证失败，Samba 将试图采用 user 安全模式。

④ domain（域）：从客户端的观点看，此模式也与 user 安全相同。只有当 Samba 服务器添加到 Windows NT 域中时（使用 smbpasswd 命令）才能被使用。当客户端试图使用这种模式连接到 samba 服务器时，用户名和密码被传输到 Windows NT 的主域或备用域控制器上。这和用 Windows NT 服务器来验证是相同的，但仍需要有效的 Linux 账户。

map to guest= VALUE：VALUE 可取值有 Never、Bad User、Bad Password 和 Bad Uid。

① Never：拒绝密码不正确的用户。

② Bad User：拒绝密码不正确的用户，但若用户不存在，则视为 guest。

③ Bad Password：将密码不正确的用户视为 guest。

④ Bad Uid：非有效的系统用户登录成功，映射为 guest（仅用于 domain 安全模式）。

encrypt passwords=yes|no：设置用户口令加密，默认为 yes。对于 domain 安全来说，这个值必须是 Yes。

update encrypted=yes|no：更新密码。可使明文密码自动转换为加密密码。该选项默认为 no，若要打开此选项则应该关闭 encrypt passwords 选项。

Obey PAM restrictions=yes|no：遵循 PAM 限制。如果使用 PAM 管理账户会话，则需要打开此选项。如果启动加密密码，则此选项不起作用。

passwd program=/usr/bin/passwd：密码程序。指出使用哪个程序来修改 Linux 用户密码。默认为/usr/bin/passwd，可用命令 which passwd 查询 passwd 命令的位置。

passwd chat=string：控制 smbd 和/usr/bin/passwd 之间的会话，用于对用户口令进行修改。

pam password change=yes|no：PAM 密码修改。若为 yes，用户要求更改口令时使用 PAM，而不用 passwd program 参数所指定的程序。

username map=/etc/samba/smbusers：用户名映射。设置用于映射 Samba 用户名的文件，默认为/etc/samba/smbusers。

UNIX password sync=yes/no：UNIX 密码同步。如果此选项打开，在 SMB 密码被修改时，Samba 试图同步更新用户的 Linux 用户密码，默认为 yes。

guest account=nobody：客户账户。设置客户账户的用户名，默认为 nobody 或 nfsnobody。当 guest ok=yes 时，可用此账号无密码访问服务器，此时的用户身份为 nobody。

guest ok=yes|no：guest 启用，默认为 no。是否允许 guest 用户视具体共享情况而定。

hosts allow =hosts：被允许的主机，其值为由逗号或白空格分隔的主机或网络，默认任何主机均可访问。

hosts deny =hosts：被拒绝的主机，格式同 hosts allow，默认不拒绝任何主机。

（4）打印机选项。

printcap name=/etc/printcap：设置 printcap 文件路径。

load printers=yes|no：设置是否自动加载打印机。

printing=cups：设置打印机系统类型。

2）共享节

（1）共享节的一般结构。

在[global]节可设置的选项是具有全局性的，这些设置后的选项可在共享节中被重新设置，但只对本节有效。要定义共享节，首先要为共享节定义一个名字，比如[printers]、[public]等，名字可以根据需要自行设定。共享节的通用格式为：

```
[sharename]                          ；节名
    comment=Remark                   ；注释。用于描述共享目录的字符串
    path=PATH                        ；路径。用于指定共享目录的路径名
    guest account= user              ；guest 账户。用于指定客户用户，一般为 nobody
    guest ok=yes|no                  ；是否允许 guest 用户。不需要密码就可以访问该共享
    read only=yes|no                 ；只读。用于指定是否只允许只读方式访问
    hosts allow=HOSTs                ；被允许的主机
    hosts deny=HOSTs                 ；被拒绝的主机
    browseable=yes|no                ；是否可以浏览
```

以上参数不一定都是必需的，必要时还可根据需要再添加一些。

（2）设置用户的家目录共享。

设置用户的家目录共享会使用[homes]节。[homes]节允许每个本地用户在其他 OS 上，比如 Windows 系统，通过 samba 访问自己的家目录，并可在自己的家目录内进行写操作。如果还有其他共享目录的话，本地用户还能同时看到它们。但是，其他用户将无法看到或共享自己的家目录。以下是一个[homes]节的设置示例。

```
[homes]
comment=Home Directories          ；设置描述串
browseable=no                     ；不允许浏览
writable=yes                      ；可写
valid users=%S                    ；所有的用户都可以通过 Windows 访问其家目录
directory mode=0775               ；目录创建权限为 0775
```

说明：在[homes]节中如果是允许 geust 用户访问的，则所有用户的家目录均可被所有客户在不提供密码的情况下看到，如果是有意这样做的话，应该取消 writable=yes，并增设 read only=yes，以起到只读作用。browseable 会从[global]中继承下来，设置 browseable=no 可以禁止其他用户看到自己的家目录。

（3）设置打印机共享。

通过设置打印机节[printers]可让指定打印机共享出去。本节的结构与[homes]基本相同，但是针对打印机的，所以出于用途的考虑，与[homes]节的设置有所不同，大致结构如下：

```
[printers]
comment=All Printers              ；设置描述串
path=/var/spool/samba             ；设置缓冲队列位置
browseable=no                     ；不允许浏览
guest ok=yes                      ；启用 guest 用户
writable=no                       ；不允许写
printable=yes                     ；允许打印
```

说明：printable=yes 是必需的，否则不能打印；path 必须指定一个每人均可写且具有 sticky 权限的目录；如果不设定打印机名，则使用本地打印机。

3．一个共享节设置示例

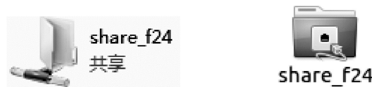
任务：设置一个共享节，可让有效用户和匿名用户均可访问。当有效用户访问时，具有读写权限；当匿名用户访问时，只有读权限。

要设置这么一个共享节，还要在 global 节中添加一个“map to guest = bad user”配置项。以下内容是一个定义在 fedora 24 上，名为 share_f24 的共享节，可满足此任务的要求。

```
[global]
        workgroup = WORKGROUP
        security = user
        hosts allow = 127. 192.168.137.
        map to guest = bad user       ；匿名用户无须交互输入用户名和密码
        guest account = nobody        ；匿名用户映射为 guest 用户
；        打印机配置忽略
```

```
[share_f24]                              ; 共享名
        comment = smb share doc in f24
        path = /var/smb_share_f24        ; 共享目录
        guest ok = yes                   ; 允许匿名用户
        read only = no                   ; 非匿名用户可写
        browseable = Yes                 ; 可以被浏览
;        valid users = usr1 usr2 ...     ; 有效的用户
;        invalid users = usr1 usr2 ...   ; 无效用户和组
;        read list = usr1 usr2 ...       ; 当 read only = no 时，有只读权限的用户
;        write list = usr1 usr2 ...      ; 当 read only = yes 时，有写权限的用户
;        hosts deny=HOSTs
```

当从 Windows 和 Ubuntu 中访问该共享时，看到的图标如下所示。



4．配置文件正确性的检查

testparm 是一个用于检查 Samba 配置文件 smb.conf 正确与否的简单程序。如果经过它检查而没有发现配置错误，则可以确实保证 Samba 服务器能够启动起来，但并不能保证服务能按所期望的那样工作。testparm 的常用方法为：

```
testparm [-s] [-h] [conf_file] [host IP]
testparm -v -s | grep usershare
```

选项-h 用于显示帮助信息；-s 用于去除测试显示与配置信息显示间的停顿及回车提示；conf_file 用于指定配置文件，若不指定则使用/etc/samba/smb.conf；host 和 IP 对用于指定主机和其上的 IP 是否允许访问 Samba 服务器。还可以使用--parameter-name pname 和--section-name sname 分别显示指定的参数配置和节配置。比如：

```
#testparm                                #显示配置文件/etc/samba/smb.conf 中的配置
#testparm -s --section-name homes        #显示 homes 节内容
```

如果 testparm 检查结果正确则返回 0，否则返回非 0 值。

5．添加 Samba 用户

当采用 user 级的安全模式时（Security=user），需要每个访问 Samba 的用户指定一个 Linux 账号，这种账号必须事先已经存在，若不存在则必须创建。设 smbshare（系统）用户已经存在，为使用其能够访问 Samba 共享资源，需要为其设置 smb 口令，命令如下：

```
#smbpasswd smbshare
```

若需多个用户，要对不同用户重复执行 smbpasswd 命令。与以前版本不同的是，默认时，用户及密码信息被存放在/var/lib/samba/private/passdb.tdb，而不再是/etc/samba/smbpasswd。

设置完成后，用户可登录 Linux 系统，并可重新运行 smbpasswd 命令来修改自己的密码。

6．Windows 端的配置

若在 Linux 和 Windows 主机之间实现信息共享，还需对 Windows 主机进行配置，大致内容如下。

（1）系统设置。主要是添加对 netBIOS 协议的支持。需要添加 NetBIOS 协议：NWLink NetBIOS 和 NWLink IPX/SPX/NetBIOS Compatibale Transport Protocol。

（2）目录设置。创建一个新目录或选择一个已经存在的目录，将其设置为网络共享，至于以什么方式共享（比如只读、读写等）由用户根据需要设定。

（3）设置允许使用共享的用户。可以使用共享的用户可以是所有用户、特定用户等，可根据具体需要设定。

18.2.4　配置共享打印机

Samba 服务提供强大的打印共享功能，可以为用户提供良好的打印服务。在默认情况下，Samba 的打印服务是开放的，只要把打印机安装好，客户端的用户就可以使用打印机。

1．建立打印机服务器

要使用本地打印机作为 Samba 共享打印机，必须首先使用打印机管理方法创建本地打印机（参见 8.2 节）。

在 Samba 服务器的主配置文件 smb.conf 中，有一些与打印机及共享相关的配置参数，如表 18-3、表 18-4 所示。

表 18-3　smb.conf 文件中常用的全局打印配置参数

参　　数	说　　明	举　　例
load printers	是否加载打印机配置文件	load printers=yes
printcap name	设置打印机配置文件路径	printcap name=cups
printing	设置打印系统类型	printing=cups

表 18-4　smb.conf 文件中常用的打印共享配置参数

参　　数	说　　明	举　　例
path	指定打印机队列位置	path=/var/spool/samba
printable	指定用户是否可打印（默认为不可以）	printable=yes
printer admin	设置打印机管理员	printer admin=root

建立打印服务器，需修改 smb.conf 或向其中添加关于打印机配置的内容，大致步骤如下。

（1）设置 global 配置项。

修改 smb.conf 的全局配置，开启打印共享功能，如下所示：

```
[global]
load printers=yes
printing=cups
printcap name=cups
```

（2）设置 printers 配置项。

```
[printers]
comment=ALL Printers
path=/var/spool/samba
browseable=no
public=yes
guest ok=yes
writable=no
printable=yes
printer admin=root
```

需要提醒的是，printable 一定要设置成 yes；path 字段定义打印机队列，可以根据需要自行设定。另外，共享打印与共享目录不同，安装完打印机之后，必须重新启动 Samba 服务，否则，客户端可能无法看到新配置的打印机。

为了使网络中的 Windows 主机能使用共享的打印机，需要安装共享打印机驱动程序。为此要首先准备驱动程序。根据上面的配置，可以使用 cupsaddsmb 命令将打印机驱动程序放入指定的/etc/samba/drivers 目录，命令如下：

```
#mkdir /etc/samba/drivers                #首先创建/etc/samba/drivers 目录
#cupsaddsmb -a -U root                   #然后运行 cupsaddsmb 命令
```

2．使用共享打印机

在 Windows 中，使用 Samba 共享打印机有多种方法，下面介绍两种常见的方法。

（1）通过“网络邻居”访问。

当配置好 Samba 共享打印机后，合法的用户可以通过“网上邻居”登录 Linux 系统，并看到共享的打印机。双击共享打印机，在弹出的窗口中确认安装此打印机的驱动程序，完成后即可使用。

（2）添加为“网络打印机”。

在 Microsoft Windows 桌面上依次选择“开始”→“控制面板”→“打印机和其他硬件”→“添加打印机”，单击“下一步”按钮，打开“添加打印机向导”对话框。选择“网络打印机或链接到其他计算机的打印机”，单击“下一步”按钮，选择要使用的打印机，在“浏览打印机”对话框中单击“下一步”按钮，系统将搜索网络中的打印机，从“共享打印机”列表中选择“共享打印所在主机名”，在“打印机”文本框中输入共享的打印机名，单击“下一步”按钮完成操作。

18.2.5　Samba 共享服务使用

Samba 服务器的资源可在服务器端使用 Samba 管理工具进行管理。用户还可在终端上通过命令来使用和检查服务器所共享的资源和状态。

1．lmhosts 文件

Samba 使用/etc/samba/lmhosts 文件存放 NetBIOS 名与 IP 地址的静态映射表，它是一个文本文件，功能类似于/etc/hosts 文件用于名字解析，结构也类似/etc/hosts，但只能有前两列。lmhosts 在 Windows 系统中名为 lmhosts.sam，所在位置一般是[C:]\windows\system32\drivers\etc。

2．Windows 客户端访问 Samba 共享

Windows 用户可通过网上邻居访问 Samba 服务器的共享目录，或在“开始”菜单中选择“运行”（Windows 键+R），在弹出的运行对话框中输入“\\Samba 服务器 IP 地址或计算机名”进行访问。

3．Linux 客户端访问 Samba 共享

（1）smbclient 命令。

smbclient 是 Samba 提供的一个类似 FTP 客户程序 ftp 的 Samba 客户程序，是一个在命令行界面下访问 Samba 共享的强有力工具。当然，在使用之前，首先要安装 Samba 的客户端。

smbclient 命令功能强大、使用复杂、参数和选项众多，这里只介绍它作为客户端命令工具的几种常见使用方法：

```
smbclient [-?|-h]
smbclient -L smbHost [-Uuser[%passwd]]
smbclient //smbHost/sharename [-Uuser[%passwd]]
smbclient -c "cmd"//smbHost/sharename [-Uuser[%passwd]]
```

这里的 smbHost 可以是主机名或 IP 地址；sharename 为共享节名，而非共享节中指定的目录名；user 为用户名，passwd 为用户 user 的密码；方括号内的内容是可选的，若不提供则在命令执行过程中提示用户输入。

“smbclient [-?|-h]”用于显示帮助信息，在不同的版本间有差异。

“smbclient -L smbHost [-Uuser[%passwd]]”用于查询服务器的共享资源。比如：

```
#smbclient -L 192.168.56.10 -Uzhang%56789
```

“smbclient //smbHost/sharename [-Uuser[%passwd]]”用于以用户 user 身份和密码 paaswd，访问共享资源 sharename。若用户名和密码验证通过，则 smbclient 会进入交互方式，提示符为“smb:…\>”，用户可以在提示符下输入类似 ftp 字符客户端的命令，比如 ls、cd、lcd、get、mget、put、mput、mkdir 和 rmdir 等命令，还可以通过?或 help 获得帮助。工作结束后，通过命令 exit 或 Ctrl_D 退出。

“smbclient -c cmd //smbHost/sharename [-Uuser[%passwd]]”是上一种形式的简化形式。当执行的命令较少时，则不需要进入交互界面，只需要把 smbclient 的命令放在-c 的参数 cmd 内就行了。若命令不只一条，可像 shell 脚本一样，用分号“;”连接起来。当 cmd 内有白空格时，需要使用引号括起来。比如：

```
#smbclient -c"ls" //192.168.56.20/public -Usomeone%123456          #列目录
#smbclient -c"get x.c" //192.168.56.20/public -Usomeone%123456     #下载
```

（2）使用 mount 命令安装 Samba 共享目录。

从用户使用 smbclient 命令访问 Samba 服务器上的共享资源情况看，较为烦琐。一种有效的方法是将 samba 的共享目录安装到本地，作为本地文件系统的一部分，之后可以像访问本地文件系统一样访问 Samba 共享资源。使用 mount 命令安装 Samba 共享文件系统的格式为：

```
mount -t cifs //smbHost/sharename mount_point
```

这里的 cifs 为 Samba 文件系统的类型。

设有一台 Windows XP 主机，机名为 win，在其上有一个名字为 study 的共享文件夹。在 Linux 系统中，将其安装在/mnt/smb/并使用它的大致过程如下：

```
#mkdir -p /mnt/smb                          #创建安装点。若已有则不需要再创建
#mount -t cifs    //win/study /mnt/smb
#ls /mnt/smb                                #列目录
#cp /mnt/smb/* /tmp                         #将其中的所有文件复制到系统临时目录
#umount /mnt/smb                            #使用完毕后，卸载之
```

（3）图形界面客户端。

还可以从桌面依次选择“Places”→“[Browse] Network”打开 Samba 客户端图形界面；或采用某种办法（比如，双击“Home”）打开“Files（文件）”资源管理器，从左侧栏选择“[Browse] Network”可以找到共享资源。之后，用户可以按照图形界面的操作方法，使用鼠标操作进行复制、粘贴和拖曳操作，这对于非专业用户来说是一个不错的选择。

（4）检查 Samba 的共享状态。

命令 smbstatus 可用来检查服务器上 samba 共享资源的状态，其用法为：

```
smbstatus [-b] [-d <level>] [-v] [-L] [-p] [-S] [-u <user>]
```

参数-b 用于简要显示状态信息；-d 用于指定调试级别；-v 用于显示冗余的输出；-L 用于显示锁信息；-p 用于显示 smbd 进程信息；-S 用于显示共享资源信息；-u user 用于显示与用户 user 相关的信息。

```
#smbstatus -b                  #简要显示 Samba 资源的使用情况
```

18.2.6　Samba 图形界面配置

对于提供 system-config-samba 软件包的系统，在安装 system-config-samba 软件包之后，可以在 CLI 界面执行 system-config-samba 命令，打开如图 18-2 所示的 samba 服务器配置主界面。

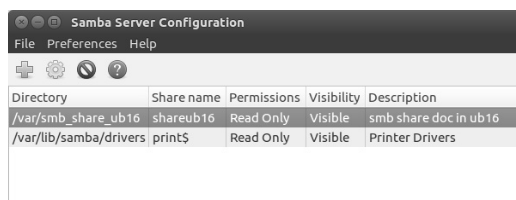


图 18-2　Samba 服务器配置主界面

通过“Preferences”→“Server Settings”可以设置[global]节中的全局变量。该界面共有

2 个选项卡（如图 18-3 所示）。通过“Basic”选项卡可以设置工作组和服务器描述串；通过“Security”选项卡可以配置安全模式和 Guest 用户。

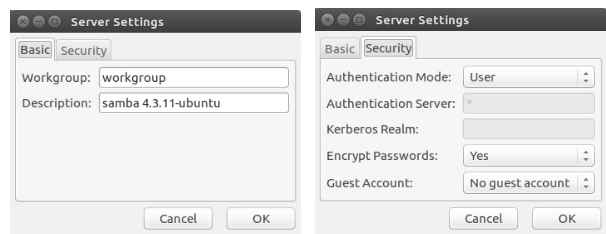


图 18-3 “Server Setting”界面

通过“Preferences”→“Samba Users”可以进行 Samba 用户管理，包括添加、修改和删除用户 3 种功能（如图 18-4 所示），在添加或修改功能中还可以为用户设置或修改密码。



图 18-4 Samba 用户管理界面

通过“File”→“Add Share”（或“+”）和“Properties”功能或相应按钮，分别可以添加或修改共享目录（界面如图 18-5 所示）；通过“Delete”可以删除选中的共享目录。

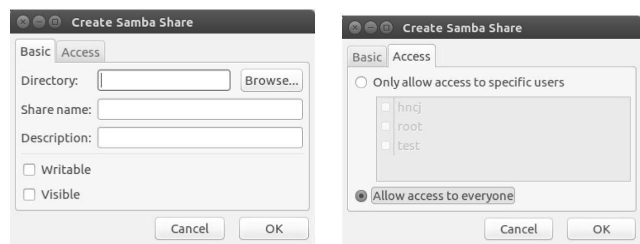


图 18-5 Samba 共享管理界面

18.2.7 关于防火墙和 SELinux 的说明

1．防火墙设置

防火墙要允许 Samba 服务所使用的端口或服务。

1）friewalld

```
#firewall-cmd --permanent --add-service samba                  #开放 samba
#firewall-cmd --permanent --add-service samba-client           #开放 samba- samba-client
#firewall-cmd --list-service                                   #检查开放的服务
```

2）ufw

```
#ufw allow samba                                    #开放 samba
#ufw status                                         #检查开放的服务
```

2．SELinux

可通过命令

```
#getsebool -a | grep -E '(smbd)|(samba)'
```

得到全部与 Samba 相关的 SELinux 布尔变量，其中常见的布尔变量及意义如表 18-5 所示。

表 18-5　samba 与 SELinux 相关的布尔变量及意义

布 尔 变 量	意　　义
samba_create_home_dirs	允许 Samba 独立创建家目录
samba_domain_controller	允许 Samba 作为域控制器，并赋予执行 useradd、groupadd 和 passwd 等命令的权限
samba_enable_home_dirs	允许共享用户的家目录
samba_export_all_ro	允许目录或文件只读共享
samba_export_all_rw	允许目录或文件读写共享
samba_run_unconfined	允许执行非限定性脚本
samba_share_nfs	允许共享 nfs 卷
smbd_anon_write	允许 smbd 写有 public_content_rw_t 属性的公共目录

在默认情况下，这些布尔变量均是未开放的，用户可以根据需要开放它们中的一些，以便 Samba 服务器与客户端都能工作。比如：

```
#setsebool -P samba_enable_home_dirs   1
#setsebool -P samba_run_unconfined   1
#setsebool -P samba_export_all_rw   1
```

18.3　NFS 服务器

18.3.1　NFS 介绍

1．NFS

UNIX 和 Linux 系统都支持 NFS，以实现系统间的数据共享。

NFS 是 Network File System 的缩写，即网络文件系统。它是由 Sun 公司开发，并于 1984 年推出的一个 RPC 服务系统，目前已经成为文件服务的一种标准，其最大的功能是可以通过网络实现文件共享。它只用于 UNIX、Linux 及类 UNIX 间的文件共享。当用户想使用远程文件时，用“mount”命令就可把远程文件系统挂载到本地文件系统的某个目录下，作为本地系统的一部分来使用。NFS 有属于自己的协议和端口号，但在传送资料或者其他相关信息时，NFS 服务器使用 RPC 协议来协助自己运行。

2．RPC

RPC（Remote Procedure Call）是远程过程调用协议。RPC 采用客户机/服务器模式。客户端发送一个有关进程参数等调用信息到服务进程，然后等待应答信息。在服务端，当收到一个调用信息后，获得进程参数并开始计算，然后将结果发送给客户端。

因为 NFS 支持的功能很多，每当启动一个功能就要启用一些端口来传输数据，因此 NFS 的功能所对应的端口是不固定的，而是通常采用随机取用一些未被使用的小于 1024 的端口作为传输之用，而客户端必须知道服务器的相关端口才能连接，此时需要 RPC 服务。RPC 的功能是制定每个 NFS 功能对应的端口号，并且回送给客户端，让客户端连接到正确的端口上。当 NFS 服务器启动时会随机启用数个端口号，并主动向 RPC 注册，这样 RPC 就知道每个端口号对应的功能了，然后 RPC 固定使用 111 号端口来监听客户机的请求并回应正确的端口号。因此，启动 NFS 之前，要先启动 RPC，否则 NFS 会无法向 RPC 注册。另外，重新启动 RPC 时原本注册的数据会丢失，因此 RPC 重新启动后，它管理的所有程序都需要重新启动以重新向 RPC 注册。

rpcinfo 命令用于显示系统的 RPC 信息，一般使用-p 参数列出某台主机的 RPC 服务。用 rpcinfo -p 命令检查服务器时，应该能看到 portmapper、status、mountd、nfs 和 nlockmgr。用该命令检查客户端时，应该至少能看到 portmapper 服务。

3．NFS 的工作原理

NFS 是一种能使服务器上的信息被其他计算机挂载而达到资源共享的网络文件系统。一个客户机可以从服务上挂载一个目录到本地文件系统作为本地文件系统的一部分。一个客户机可以是 NFS 服务器或 NFS 客户机，甚至可以同时为 NFS 服务器和客户机。

只有服务器所共享出去的文件或目录，NFS 客户机才能够挂载它们。NFS 服务所能共享出去的目录都记录在/etc/exports 文件中，当 NFS 服务器启动时，会自动启动 exportfs 程序，搜索/etc/exports 文件，并且给所有共享出去的文件或目录赋予正确的权限。

NFS 有多个守护进程，其中的 rpc.nfsd 和 rpc.mountd 的功能如下所述。

（1）rpc.nfsd：基本的 NFS 守护进程，主要功能是管理客户端是否能够登录服务器。

（2）rpc.mountd：RPC 安装守护进程，主要功能是实施 NFS 协议、管理实施 NFS 文件系统。主要功能是输出 NFS 文件系统和管理 rmtab 文件。当收到客户端挂载请求时，它会检查 NFS 共享文件列表及对应的访问控制权限。当访问者有权限访问时，就为用户提供服务。

除此之外，还有 rpcbind，主要功能是进行端口映射工作。当客户端尝试连接并使用 RPC 服务器提供的服务（如 NFS）时，它会将所管理的与服务对应的端口提供给客户端，从而使客户机可以通过该端口向服务器请求服务。nfs 的端口是 2049，但是它基于 rpcbind 的（端口是 111）。

18.3.2　NFS 文件系统配置

1．软件安装

红帽和 Ubuntu 系统所使用的软件包分别为 nfs-utils 和 nfs-kernel-server，当然还都会需要

其他包的支持，比如 rpcbind 等，这些支持包会在 NFS 包安装时被自动安装，安装方法如下：

```
#yum install nfs-utils                                    #红帽系统
#apt install nfs-kernel-server                            #Ubuntu 系统
```

2．服务管理

NFS 的服务为 nfs-server.service，可以使用服务管理的办法管理它的启动，方法如下：

```
#systemctl enable/disable/status nfs-server.service       #启用/禁用
#systemctl status nfs-server.service                      #检查状态
#systemctl start/stop/restart/reload nfs-server.service   #启动/停止/重启/重载
```

说明：出于对安全的考虑，NFS 服务默认是关闭的，可在使用前开启它；每当 NFS 配置文件被修改后，都要重启或重载 NFS 服务，建议方法是重载。

3．NFS 服务器端的配置

NFS 配置文件为/etc/exports。在红帽系统中，还可使用附加的配置文件/etc/exports.d/*.exports，前者为主配置文件，后者为附加的配置文件，两者结构相同。

配置文件/etc/exports 是一个文本文件，空行无效，由“#”引导的行为注释行。每一有效行描述一个共享目录，并且说明该目录如何被共享。每一有效行的格式如下：

共享目录 访问者(选项) 访问者(选项)……

“共享目录”与“访问者”间用空格分隔，“访问者”为主机或 IP 地址，“访问者”与“(选项)”间无分隔符，选项若不止一个则用逗号分隔。共享目录以绝对路径方式给出，括号内的选项只对括号前的访问者有效，访问者的格式可以是：

（1）指定 IP 地址的主机，如 192.168.137.2。

（2）指定网段中的所有主机，如 192.168.137.0/24，192.168.56.0/255.255.255.0。

（3）单台主机名，如 pc001。

（4）含有通配符的主机名，如 proj*.local.domain，表示 local.domain 域的所有以 proj 开头的主机。

常用的选项及说明如表 18-6 所示。

表 18-6 /etc/exports 文件中常用选项及说明

选项	说明
ro/ rw	只读/读写
sync/ async	同步/异步写入
secure	只允许 NFS 通过 1024 以前端口连接（默认）
insecure	允许 NFS 通过 1024 及以上的端口连接，此选项关闭 secure
secure_locks	使用安全的文件锁
insecure_locks/no_auth_nlm	允许使用不安全的文件锁。此选项关闭 secure_locks
wdelay	延迟写。如果多个用户要写入 NFS 目录，则归组写入（默认）
no_wdelay	同步写入。当使用 async 时，无须此设置
hide	不共享子目录或隐藏子目录

（续表）

选　　项	说　　明
no_hide	共享 NFS 目录的子目录
subtree_check	检查共享目录树。如果共享/usr/bin 之类的子目录时，强制 NFS 检查父目录的权限（默认）
no_subtree_check	与 subtree_check 相对，不检查父目录权限
root_squash	客户机用 root 用户访问该共享文件夹时，将 root 映射成 nfsnobody 用户（默认）
no_root_squash	与 root_squash 相对。客户机用 root 访问该共享文件夹时，不做映射，主要用于无盘客户机
all_squash	把客户机上的所有用户映射为匿名用户（nfsnobody），适合公用目录
anonuid=xxx, anongid=xxx	使用 all_squash 选项时，指定 NFS 服务器/etc/passwd 文件中用户的 UID 和 GID
no_all_squash	保留共享文件的 UID 和 GID（默认）

4．配置示例

示例 1：

```
/var/nfs_share    zhang(rw)    wang(rw,no_root_squash)
```

该行表示共享服务器上的/var/nfs_share 目录，只有 zhang 和 wang 两台主机可以访问，并且两台主机对该共享目都有读写权限；主机 zhang 在用 root 访问时，将客户机的 root 用户映射成服务器上的匿名用户（root_squash 参数为默认参数），相当于在服务器上使用匿名用户（nfsnobody）访问该目录；wang 主机在用 root 用户访问该共享目录时，不映射 root 用户（no_root_squash 参数），即相当于在服务器上用 root 身份访问该目录。

示例 2：

```
/projects proj*.UNIX.gjshao(rw)
```

该行表示共享目录为/projects，UNIX.gjshao 域中所有以 proj 开头的主机都可以访问该目录，并且都有读写的权限；客户机上的任何用户在访问时都映射成匿名用户（没有指定 no_all_squah 参数，默认 all_squash）。

示例 3：

```
/var/nfs_share 192.168.137.0/255.255.255.0（rw,all_squash）   192.168.200.0/24（ro）
```

该行表示共享目录为/var/nfs_share，192.168.137.0/24 网段的所有主机都可以访问该目录，且对该目录有读写的权限；192.168.200.0/24 网段的所有主机对该目录有只读权限，在访问时所有的用户都映射成匿名用户。

5．配置 NFS 客户端

NFS 客户端需要软件包 cifs-utils 的支持，使用前需要安装。

客户端对 NFS 的使用，表现在使用 mount 命令将服务器上的共享目录安装到本地，而作为本地系统的一部分，使用方法如下：

```
#mount -t nfs [options] server:dir    mount_point
```

其意义是将 server 服务器上的 dir 目录安装在本地的 mount_point 安装点上，文件系统类型为 nfs。比如：

```
#mount -t nfs -r ubuntu16:/var/nfs_share    /mnt/nfs
```

将 ubuntu16 上/var/nfs_share 目录以只读方式安装在本地的/mnt/nfs 目录。

18.3.3　NFS 系统的使用

1．使用示例

设某单位需要在网络上共享一个文件夹/var/nfs_share，让所有人都只有只读权限，且只有 192.168.137.0/24 子网的用户可以访问，试通过 NFS 共享该文件夹。设 NFS 服务器的 IP 地址为 192.168.137.5，大致操作步骤如下。

（1）建立共享目录。

以 root 身份登录，检查/var/nfs_share 是否存在，若不存在则创建，方法是：

```
#mkdir -p /var/nfs_share
```

然后再向其中添加一些用于共享的文件，比如：

```
#cp /etc/exports /etc/init.d/*    /var/nfs_share
```

（2）编辑/etc/exports 文件。

在/etc/exports 文件中加入如下行：

```
/var/nfs_share 192.168.1.0/24(ro,all_squash)
```

（3）重载 NFS 服务。

```
#systemctl reload nfs-server
```

（4）安装 NFS 文件系统。

在 192.168.137.0/24 子网上的任意一台主机上安装共享目录：

```
#mkdir -p /mnt/nfs
#mount -t nfs 192.168.137.5:/var/nfs_share /mnt/nfs
```

将 192.168.137.5 上的/var/nfs_share 目录挂接到本机的/mnt/nfs 目录下。

（5）测试权限。

可通过命令 ls -l 查看其中的内容

```
#ls -l /mnt/nfs
```

当执行命令“mkdir /mnt/nfs/test”，在其中创建目录 test 时，将得到以下错误提示：

```
mkdir: cannot create directory `/mnt/nfs/test': Permission denied
```

这是因为在步骤（2）中的配置中使用了 ro 且有 all_squash 的缘故，任何用户对该目录的共享都只有读权限，且用户被映射为 nfsnobody。

（6）拆卸。

NFS 文件系统的拆卸与普通文件的拆卸方法相同，方法为：

```
umount /mnt/nfs
```

2．NFS 系统常用命令

（1）exportfs 命令。

exportfs 命令使 NFS 服务器重新读取/etc/exports 文件中的设置。该命令可以在改变 exports 文件配置之后使之立即生效，而不需要重新启动 NFS 服务器，相当于重载服务器。命令用法为：

```
exportfs [-airuv] [-o options, …] [client:/path …]
```

其常用参数如表 18-7 所示。

表 18-7　exportfs 的常用参数

参　　数	意　　义	参　　数	意　　义
-a	输出或不输出所有文件	-i	忽略/etc/exports 文件，只使用指定或默认选项
-r	重新输出所有的信息	-u	卸载目录
-v	显示冗余信息	-o opts	指定输出选项

（2）showmount 命令。

在正确设置了 NFS 共享目录并正确启动 NFS 服务器后，可以使用 showmount 命令查询 NFS 的共享状态，命令格式为：

```
showmount [-adehv] [--all] [--directories] [--exports] [host]
```

其常用参数如表 18-8 所示。

表 18-8　showmount 的常用参数

参　　数	意　　义	参　　数	意　　义
-h|--help	帮助信息	-a|--all	显示客户端与被挂载的目录信息
-d|--dirs	只显示被客户端安装的目录信息	-e| --exports	显示服务器的 export list

18.3.4　NFS 的其他功能

1．NFS 自动挂载和自动卸载

当使用 NFS 共享文件时，需要首先挂载，挂载后要一直在线，只要任一方离线，都会造成另一方等待超时，那么，有没有一个方法能让用户使用时自动挂载，不使用时自动卸载？有。可以使用 autofs 服务，它使用 automount 守护进程来管理挂载点，使得文件系统只在被访问时才被动态地安装，当一段时间不使用的话，又会被自动卸载。Autofs 使用主配置文件/etc/auto.master 来决定要定义哪些挂载点，然后，它使用适用于各个挂载点的参数来启动 automount 进程。

Linux 的 autofs 软件包为 autofs，若使用前需要安装它，方法如下：

```
#yum install    autofs                  #红帽系统
#apt install    autofs                  #Ubuntu 系统
```

1）主配置文件

autofs 的主配置文件/etc/auto.master 中每一有效行定义一个安装点，其结构为：

```
mount_point    map_file    [options]
```

其中 mount_point 为安装点，map_file 为本安装点对应的配置文件，options 为可选的，这里不使用它。在 autofs 软件包安装时，也同时为我们安装了样本配置文件/etc/auto.master 和/etc/auto.misc。比如：

```
/misc /etc/auto.misc
```

就是样本文件中的一行，定义一个安装点/misc，安装方法在/etc/auto.misc 中描述。也可以在其中定义我们自己的一行，比如：

```
/nfs    /etc/auto.nfs
```

用来自动安装 nfs 文件系统。为了描述方便，我们把安装方法描述文件/etc/auto.misc 称为自动安装映射文件。

2）自动安装映射文件

自动安装映射文件的结构为：

```
key    [-options]    location
```

key 为主配置文件内安装点下的子目录，该子目录可存在，也可以不存在；-options 为安装参数；location 为要安装的文件系统的位置。autofs 样本配置文件/etc/auto.misc 的内容如下。

```
#This is an automounter map and it has the following format
#key [ -mount-options-separated-by-comma ] location
#Details may be found in the autofs(5) manpage
cd                -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom
#the following entries are samples to pique your imagination
#linux            -ro,soft,intr              ftp.example.org:/pub/linux
#boot             -fstype=ext2               :/dev/hda1
#floppy           -fstype=auto               :/dev/fd0
#floppy           -fstype=ext2               :/dev/fd0
#e2floppy         -fstype=ext2               :/dev/fd0
#jaz              -fstype=ext2               :/dev/sdc1
#removable        -fstype=ext2               :/dev/hdd
```

其中示范性地描述了文件系统及位置。针对 NFS 文件系统自动安装问题，只要文件的尾部增加如下一行：

```
nfs        -rw        ub16clt1. UNIX.gjshao:/var/nfs_share
```

就可将 ub16clt1. UNIX.gjshao 上的/var/nfs_share 自动安装到本地系统下的/misc/nfs。

3）启动 autofs 服务

autofs 服务为 autofs.service，管理方法为：

```
#systemctl enable/disable/status/start/stop/reload autofs.service
```

若 autofs 已经启用且正在运行，则当配置好/etc/auto.master 和/etc/auto.misc 文件后，只要

重启 autofs 服务，就可以自动安装 NFS 文件系统，方法如下：

```
#systemctl reload autofs
```

当用户首次访问/misc/nfs 时，文件系统就会被自动安装，当 5min 不使用时 automount 就会自动卸载它。

2．NFS 服务器的故障排除

NFS 一些常见的错误信息及错误描述如表 18-9 所示。

表 18-9　NFS 错误信息及描述

错误信息	描　　述
Too many levels of remote in path	重复挂载：试图挂载一个存在的文件系统
Permission denied	没有权限：NFS 服务器拒绝客户机挂接，或用户在服务器上不存在
No such host	主机名或地址错：通常是/etc/hosts 文件或 DNS 配置错误
No such file or directory	文件或目录不存在：通常是访问的目录不存在
NFS server is not responding	NFS 服务器不响应。可能是 NFS 过载或 NFS 停止工作
Stale file handle	文件控制失败：在客户端使用期间服务器端文件被删除
Fake hostname	假冒的主机：forward 和 reverse 的 DNS 记录在客户端已不存在
program not registered	NFS 服务器未开启

当 NFS 服务使用出问题时，应从以下几个方面着手解决：服务器和客户端之间的网络是否通；portmap/rpcbind 和 nfs 服务是否正常启动；/etc/exports 配置文件是否正确；客户端中的 mount 命令语法或/etc/fstab 的格式是否正确；服务器的负荷是否太高；内核是否支持 NFS 和 RPC 服务；防火墙规则设置和 SELinux 是否允许等。

3．使用 nfsstat 命令查看 NFS 服务器状态

nfsstat 命令用于显示关于 NFS 客户端和服务器端的活动统计信息。该命令用法简单，格式为：

```
nfsstat [options] …
```

其参数如表 18-10 所示。

表 18-10　nfsstat 的常用参数

参　　数	意　　义	参　　数	意　　义
-c|--client	显示客户端信息	-s|--server	显示服务器端信息
-n|--nfs	只显示 NFS 信息。默认显示 NFS 和 RPC 信息	-r|--rpc	只显示 RPC 信息，默认显示 NFS 和 RPC 信息
-m|--mount	显示已经安装的 NFS 文件系统信息	-2|-3|-4	仅显示版本 v2|v3|v4 信息
-v|-o opt	显示全部或由 opt 指定的信息	-l|--list	列表方式输出

表 18-10 中的-o 选项的参数 opt 可取的值有：nfs（NFS 协议信息）、rpc（RPC 信息）、net（网络层信息）和 all（所有信息），当为 all 时，-o all 相当于-v。

nfsstat 的使用示例如下：

```
#nfsstat -o all -234                    #显示所有版本信息
```

```
#nfsstat --nfs --server -3              #显示版本 server 统计信息
#nfsstat -m                              #显示已经安装的 NFS 文件系统信息
```

18.3.5　关于 NFS 的其他说明

1．nfs 与防火墙

防火墙要允许 NFS 服务所使用的端口或服务。

1）firewalld

```
#firewall-cmd --add-service nfs
```

2）ufw

```
#ufw allow nfs
```

2．nfs 与 SELinux

SELinux 通过布尔变量和类型标签对 nfs 进行控制，可通过“man nfsd_selinux”得到更详细的信息。

1）布尔变量

通过命令

```
#getsebool -a | grep -E '(cifs)|(nfs)'
```

可以得到所有与 nfs 相关的布尔变量及状态，常用的布尔变量及意义如表 18-11 所示，用户可根据实际需要打开或关闭它们。

表 18-11　nfs 与 SELinux 相关的常用布尔变量及意义

布尔变量	意义	布尔变量	意义
ftpd_use_nfs ftpd_use_cifs	允许 ftp 服务器访问 nfs 卷	httpd_use_nfs httpd_use_cifs	允许 httpd 服务器访问 nfs 卷
samba_share_nfs	允许 samba 服务器访问 nfs 卷	use_nfs_home_dirs	添加对家目录的支持
virt_use_nfs	允许虚拟 guest 用户管理 nfs 卷	daemons_use_tty	允许服务进程读写终端
nfs_export_all_ro	将共享信息输出为只读	nfs_export_all_rw	将共享信息输出为读写
nfsd_anon_write	允许匿名用户 nfsnobody 写具有 public_content_rw_t 类型的目录	daemons_use_tcp_wrapper	允许 nfs 服务使用 tcp wrapper

2）类型标签

nfsd 进程拥有 nfsd_t 类型，可以使用以下命令查询：

```
#ps -eZ | grep nfsd_t
```

其他文件或目录的 SELinux 类型及管理参见 10.5 节。

习题 18

1．思考题

（1）什么是 SMB？什么是 Samba？
（2）叙述 smb.conf 文件的作用和结构。
（3）如何设置文件目录共享？
（4）如何设置打印机共享？
（5）如何从 Linux 访问 Windows 或另一台 Linux 的共享资源？
（6）什么是 nfs？它主要用于什么情况下的共享？

2．单项选择题

（1）Samba Server 共有多种验证方式，其中验证级别最低的是（　　）。
A．域　B．服务器　C．共享　D．用户
（2）Samba Server 的默认主配置文件是（　　）。
A．samba.conf　B．smb.conf　C．lmhosts　D．printcap
（3）在 Linux 下可以使用远程挂载方法访问共享资源的命令是（　　）。
A．smbmount　B．mount　C．smbclient　D．umount
（4）在所用系统下，Samba 文件系统的类型是（　　）。
A．ext2　B．smb　C．nfs　D．cifs

实验 18

1．配置和使用 Samba 文件共享。
2．在两个 UNIX/Linux 系统间通过 nfs 实现共享。

第 19 章　域名服务器 DNS

19.1　DNS 概述

DNS（Domain Name Server，域名服务）是 TCP/IP 网络最基本和最重要的服务之一，它担负着 Internet、Intranet、Extranet 等网络的域名的解析任务，DNS 服务器的好坏将直接影响到整个网络的运行。TCP/IP 网络通过 IP 地址来识别机器与设备，在 IPv4 中它是以点分十进制的形式存在的，记忆这些纯数字的 IP 地址不仅枯燥，且容易出错，为了解决这个问题，人们使用了域名。所谓域名，就是将在计算机网络中标识一台计算机的 IP 地址用一组有实际意义、较容易记住的名称来代替。域名（Domain Name）是由一串用点分隔的名字组成的 Internet 上某一台计算机或计算机组的名称，用于标识计算机的网络位置，或者再通俗点说，域名就相当于一个家庭或单位或机构的门牌号码，其他人可由这个号码很容易地找到标识的位置。从这个意义上说，它解决了 IP 地址不便记忆的问题，让用户可以完全不必知道 IP 地址存在的情况下使用网络。因此，必须有一种机制或办法，通过它能够将域名翻译成计算机通信所需用的 IP 地址，这就是本章所要介绍的 DNS 服务器。

19.1.1　IP 与域名的转换

在 UNIX/Linux 系统中，可以采用三种技术来实现域名和 IP 地址之间的转换，它们分别是 host 表，NIS（Network Information System，网络信息服务）和 DNS。

host 表是本地数据库文件，对应于/etc/hosts 文件（见网络管理与网络应用部分），它是文本文件，可直接使用文本编辑器编辑。host 表是 DNS 服务器软件的雏形，在 DNS 出现以前，主机之间为了通过名字进行通信，一般采用 hosts 文件。每个主机的 host 表应该包含自己通信可达的所有主机信息，如果一个网内主机要求能够相互通信的话，就要求网内所有主机都应包含相同的记录，且每当网内有主机增减，管理员需要经常更新该文件，并将新的配置文件发到网上的每台主机中，随着网络中主机数量的不断增加，这种修改与同步非常烦琐，因此该方法只适用于主机数量很少的网络。

NIS 起源于 SUN Microsystems 的 Yellow Pages（YP）服务，用于共享需要在网上所有主机上使用的信息。NIS 使用基本的 C/S 架构，并且通过一个 NIS 域名进行管理区域的划分，这个域名与我们在 Internet 上使用的域名格式相同，但只是 NIS 用来标识区域的。NIS 将主机表用作数据库，使用标准的 DBM 格式存储，客户机可以在这里得到所需的主机表信息。因此，NIS 将所有的主机数据都保存在中央主机上，由中央主机将所需数据分配给所有的服务器，从而导致将主机域名转换为 IP 地址时效率很低，只适用于中型网络。

DNS 是一种采用 C/S 机制实现名称与 IP 地址转换的系统。DNS 是由“名字”分布数据库组成的，在网络中并没有一个中心数据库来存放 Internet 上的全部主机信息，这些信息分布在一个层次结构中的若干域名服务器上。它建立了叫域名空间的逻辑树结构，是负责分配、修改、查询域名的组合服务系统，该空间上的每个结点或域都有唯一的“名字”。

19.1.2　域名空间和区域

1．域名空间

因为整个 DNS 是采用分布数据库组成的逻辑树结构，如图 19-1 所示，这个树结构被称为 DNS 域名空间。图中显示了顶级域及下一级子域的树结构关系，每个结点以及其下的所有结点称为一个域，域可以有主机域和其他域。

DNS 规定，域名中的“名字”都由英文字母和数字组成，每一个标号不超过 63 个字符，也不区分大小写字母。标号中除连字符（-）外不能使用其他的标点符号。级别最低的域名写在最下边，级别最高的域名写在最上边。由多个标号组成的完整域名总共不得超过 255 个字符。在图 19-1 中，www.hncj.edu.cn 就是一个主机，而 hncj.edu.cn 则是一个子域。一般在子域中会包含多个主机，hncj.edu.cn 子域下包含 www.hncj.edu.cn、ftp.hncj.edu.cn 等多台主机。

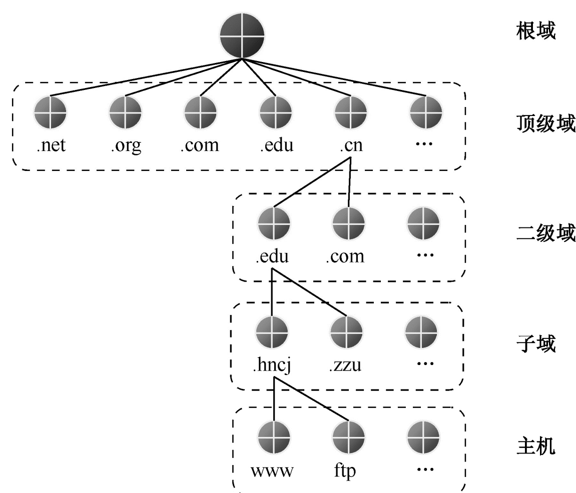


图 19-1　域名空间示意图

2．与 DNS 相关的概念

根域：代表域名命名空间的根，一般用圆点“.”来表示。

顶级域：指直接处于根域下面的域，有国际顶级域名和国家顶级域名两大类。在实际使用和功能上，国际域名与国内域名没有任何区别，都是互联网上的具有唯一性的标识。在 Internet 中，顶级域名由 InterNIC（Internet Network Information Center，国际互联网络信息中心）进行分配、管理和维护。

二级域：指顶级域之下的域。在国际顶级域名下，它是指域名注册人的网上名称，在国家顶级域名下，它是表示注册企业类别的符号。在 Internet 中，二级域也是由 InterNIC 负责管理和维护的。

子域：指二级域下面所创建的域。它一般由各个组织根据自己的需要自行创建和维护的。

主机：指域名空间中最下面一层，它被称为完全合格域名或全称域名（Fully Qualified Domain Name，FQDN）。

3．DNS 区域（zone）

为了便于根据实际情况来分散域名管理工作的负荷，DNS 服务器以 zone（区域）为单位管理域名空间。zone 是一个用于存储单个 DNS 域名的数据库，它将域名空间分成较小的区段。zone 中的数据保存在管理它的 DNS 服务器。在现有的域中添加子域时，该子域既可以包含在现有的 zone 中，也可以为它创建一个新 zone 或包含在其他的 zone 中。一个 DNS 服务器可以管理一个或多个 zone，而一个 zone 也可以由多个 DNS 服务器管理。

19.1.3　DNS 查询

当客户机需要访问 Internet 上的主机时，首先需要根据对方域名或主机名通过本地 DNS 服务器获得它的 IP 地址，这称为正向解析。在查询过程中，往往在本地 DNS 服务器不一定有对应数据，这时会继续向另外一台 DNS 服务器查询，直到解析出要访问主机的 IP 地址或失败为止，这一过程称为查询。DNS 查询有 3 种，递归查询、迭代查询和反向查询。反向查询，也叫反向或逆向查询或解析，是根据 IP 地址查询主机名或域名的过程，得到的结果可能不唯一。

1．递归查询

递归查询（Recursive Query）是最常见的查询方式，域名服务器将代替提出请求的客户机或下级 DNS 服务器进行域名查询，若域名服务器不能直接回答，则会在域内各子树的所有分支进行递归查询，最终将返回查询结果给客户机。在域名服务器查询期间，客户机将完全处于等待状态。在这种方式中，DNS 服务器必须向 DNS 客户端作出回答，要么是查询结果，要么是失败信息。一般由 DNS 客户端提出的查询请求都是递归型的查询方式，目前通常采用这种方式。

2．迭代查询

当第一台 DNS 服务器收到用户的查询请求时，如果第一台 DNS 服务器没有找到对应的信息，它会向用户返回一个最佳主机地址，也就是第二台 DNS 服务器，然后用户向第二台服务器发出查询请求，如果第二台 DNS 服务器还没有找到对应的信息，则第二台 DNS 服务器会向用户返回第三台服务器的地址。这个过程会一直重复，直到用户找到所需的数据，如果最后一台服务器中还没有找到所需的数据，则通知用户查询失败。该查询方式多用于 DNS 服务器与 DNS 服务器之间的查询。

3．反向查询

反向查询（Reverse Query）是根据 IP 地址解析对应主机名的查询。由于 DNS 名字空间中域名与 IP 地址之间无法建立直接对应关系，所以必须在 DNS 服务器内创建一个反向查询的区域，该区域名称的最后部分为 in-addr.arpa。由于反向查询占用大量的系统资源，会给网络带来不安全，因此通常均不提供反向查询。

19.1.4 客户端与域名解析相关的配置文件

客户端要想正确地获得域名解析，需要配置一些文件，与域名解析相关的文件有/etc/hosts、/etc/host.conf 和/etc/resolv.conf，详细信息参见网络管理与网络应用部分。

19.1.5 DNS 服务器的类型

DNS 服务器是用来存储主机域名信息的，可以分为以下 3 类。

1．Primary DNS

Primary DNS（主域名服务器）是一个功能完备的 DNS 服务器，管理一个或数个域（Domain）的计算机信息。这些相关的信息都是域管理员构造的本地文件，依照某种规定格式储存于服务器的指定目录中，然后会在服务器启动时将这些资料载入系统。

配置主服务器需要一整套的配置文件，包括主配置文件、正向域区域文件、反向域区域文件、缓存初始化文件和本地回送文件。一个域中只能有一个主域名服务器。

2．Slave DNS

Slave DNS（辅助域名服务器）也是一个功能完备的 DNS 服务器，所不同的是，它的主机信息并非完全储存于自己的系统中，而是由某个主域名服务器来提供。

辅助域名服务器是主域名服务器的备份，配置辅助域名服务器不需要生成本地区域文件，只需配置主配置文件、缓存文件和本地回送文件，区域文件会自动从主域名服务器复制过来。

启动区域传输机制的情况只有 3 种，一是辅助域名服务器刚启动，二是 SOA 记录中的刷新间隔到达，三是 master DNS 设置了主动通知辅助域名服务器数据有变化。以上 3 种情况之一发生后，辅助域名服务器会主动与主域名服务器进行数据同步。

3．Cache-Only DNS（唯缓存服务器）

一个 DNS 服务器不可能拥有互联网上所有的主机信息，因此它提供了一种转送方式，将自己的 DNS 服务器无法处理的查询要求转送至上一层的 DNS 服务器上查询，然后将得到的查询结果传送给提出查询要求的主机。而 Cache-Only DNS 就是这样一种服务器，在该 DNS 服务器的主机里，除了自己的信息外就没有其他的信息了，它将所有的查询要求全都转送至其他 DNS 服务器上。在网络上设置缓存服务器可以减少主域名服务器和辅助域名服务器的负担，减少网络传输。

唯缓存服务器本身不管理任何域，仅运行域名服务器软件，但没有域名数据库。对于唯缓存服务器只需要配置一个缓存文件，但通常还要配置一个回送文件。

在实际应用中，可以根据实际需要来选择合适类型的服务器并进行合理配置。建议在大型的网络中，无论运行哪种类型的域名服务器，都要配置一个唯缓存服务器。

19.2　BIND

19.2.1　BIND 简介

BIND（Berkeley Internet Name Domain）由美国加利福尼亚大学伯克利分校开发，现由互联网系统协会（Internet Systems Consortium）负责开发与维护，是现今互联网上最常用的 DNS 服务器软件，大部分 Linux 发行版都附带 BIND 安装包。

BIND 包含以下组件：一个域名服务器，一个域名系统解析器库和验证 DNS 服务器的一些工具。最新版本的 BIND 可以在 http://www.isc.org 下载。

19.2.2　安装 BIND

1．红帽系统

红帽系列系统的 DNS 服务器软件包为 bind，与之相关的软件包有：

```
bind                    #服务器软件包
bind-chroot             #chroot 运行环境支持包
bind-libs               #bind 软件包使用的库
bind-libs-lite          #bind 开发所需的 Lite 文件头文件和库
bind-license            #bind 套件许可证
bind-utils              #DNS 域名服务实用程序
```

其中 bind-chroot 是以 chroot 方式运行的 bind 版本，可以不安装。但若不安装，不能以 chroot 方式运行 bind。

可以使用以下命令检测它们是否已经安装：

```
#rpm -qa | grep -i '^bind'              #或
#yum list installed bind*
```

假设主机已接入 Internet，可以使用如下命令安装：

```
#dnf install bind bind-chroot           #安装 bind 及相关组件
```

2．Ubuntu 系统

Ubuntu 系统的 DNS 服务器软件包为 bind9，可用以下命令检查 bind9 是否已经安装：

```
#apt list --installed bind9*                  #或
#dpkg -l bind9*
```

若还没有安装，可使用以下命令安装：

```
#apt install bind9
```

19.2.3　bind 的启动管理

在红帽系统下，DNS服务器的服务名为named-chroot.service（chroot方式）或named.service（非 chroot 方式）；在 Ubuntu 系统下为 bind9.service。可以使用服务管理方法管理 DNS 服务器的运行。

1．立即启动、停止、重启动或重载配置

```
#systemctl start|stop|restart|reload|status named-chroot          #红帽系统：以 chroot 方式运行
#systemctl start|stop|restart|reload|status named                 #红帽系统：以非 chroot 方式运行
#systemctl start|stop|restart|reload|status bind9                 #Ubuntu 系统
```

这里的 reload 用于修改配置后重新装入配置参数。

2．开机自启动、取消自启动

```
#systemctl enable|disable named-chroot          #红帽系统：以 chroot 方式运行
#systemctl enable|disable named                 #红帽系统：以非 chroot 方式运行
#systemctl enable|disable bind9                 #Ubuntu 系统
```

19.2.4　DNS 服务器的运行方式及工作目录

DNS 服务器的主配置文件是 named.conf，但它所在的目录在不同的系统是不同的，所使用的配置文件也有区别。

1．Ubuntu 系统的 bind9

Ubuntu 系统的 DNS 服务器 bind9 的配置目录是/etc/bind。bind9 刚安装完毕之初，在配置目录/etc/bind 内有以下文件：named.conf、named.conf.local、named.conf.options、named.conf.default-zones、bind.keys、zones.rfc1918、rndc.key、db.root、db.local、db.empty 等。其中 named.conf 为主配置文件，其他文件会以某种方式包含到该文件中。

2．红帽系统

在红帽系统中 bind 可以 chroot 方式运行，也可以非 chroot 方式运行。

1）非 chroot 方式

当以非 chroot 方式运行时，配置文件有/etc/named.conf、/etc/named.rfc1912.zones、/etc/named.iscdlv.key、/etc/named.root.key，还有一个名誉上的配置目录/etc/named。实际的工作目录为/var/named，在其中有 named.ca、named.empty、named.localhost、named.loopback 文件和 slaves、data 等目录，这些文件和目录的作用可以从主配置文件/etc/named.conf 和其包含的文件 named.rfc1912.zones 中看到。

2）chroot 方式

当 bind 以 chroot 方式运行时，可以使用命令

```
#ps -ef | grep named
```

观察 bind 的运行方式。该命令的输出为：

```
named    …    /usr/sbin/named -u named -t /var/named/chroot
```

从中可以看到进程/usr/sbin/named 的所有者为 named，/var/named/chroot 为 DNS 服务器 bind 工作的根目录，也就是说当 named 以 chroot 方式运行时，它将被禁锢在目录/var/named/chroot/中，不能再看到该目录以外的世界。关于这一点可以通过 man named 查看 named 命令-t 参数的意义。此时，named 的工作根目录为/var/named/chroot/，其中内容有 dev、etc、run、usr 和 var 目录，这是一套相对于原系统根目录/named 所需要的目录，其中的 etc 及其内容和 var 及其内容分别为 named 以 chroot 方式运行的配置文件和工作目录。

由上可见，在红帽系统中，named 以 chroot 方式和非 chroot 方式运行时的配置文件是相同的，但配置目录和工作目录不同，分别是/etc 与/var/named 及/var/named/chroot/etc 和/var/named/chroot/var/named。主配置文件分别为/etc/named.conf 和/var/named/chroot/etc/named.conf。在 Ubuntu 系统中配置目录为/etc/bind，主配置文件为/etc/bind/named.conf。在配置 DNS 服务器时，配置方法相同，但要注意配置文件或目录的位置。

19.2.5　DNS 服务器配置基础

1．主配置文件

bind 软件包被安装后，就会在配置目录下生成一个默认的主配置文件 named.conf，这是一个唯缓存服务器配置文件。也就是说，当以此主配置文件运行 DNS 服务，这台机器就成了一个唯缓存服务器。

某红帽系统的默认主配置文件 named.conf 内容如下：

```
// named.conf
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//
options {
   listen-on port 53 { 127.0.0.1; };
   listen-on-v6 port 53 { ::1; };
   directory         "/var/named";
   dump-file         "/var/named/data/cache_dump.db";
   statistics-file "/var/named/data/named_stats.txt";
   memstatistics-file "/var/named/data/named_mem_stats.txt";
   allow-query        { localhost; };

   /* - If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.
      - If you are building a RECURSIVE (caching) DNS server, you need to enable recursion.
      - If your recursive DNS server has a public IP address, you MUST enable access
```

```
        control to limit queries to your legitimate users. Failing to do so will
        cause your server to become part of large scale DNS amplification
        attacks. Implementing BCP38 within your network would greatly
        reduce such attack surface
    */
    recursion yes;
    dnssec-enable yes;
    dnssec-validation yes;
    managed-keys-directory "/var/named/dynamic";
    pid-file "/run/named/named.pid";
    session-keyfile "/run/named/session.key";
    /* https://fedoraproject.org/wiki/Changes/CryptoPolicy */
    include "/etc/crypto-policies/back-ends/bind.config";
};

logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

由此可以看出，named.conf 文件由多个配置语句组成，每个配置语句后是参数和用大括号括起来的配置子句块。各配置子句块也包含相应的参数，并以分号结束。最常用的配置语句有 options、zone 和 include 等，可使用 man named.conf 命令来查看配置文件的详细说明。

1）options 语句

options 语句定义了全局配置选项，在 named.conf 文件中有且只能有一个，其基本格式为：

```
options    {
    配置子句;
};
```

最常用的配置子句有下列两种：

（1）directory（目录名）：定义工作目录，配置文件中所有使用的相对路径，指的都是在这里配置的目录下。默认为/var/named，通常不需要修改。以后配置 DNS 所需要创建的区域文件也要存放在此目录内。

（2）forwards {IP 地址}：定义将域名查询请求转发给其他 DNS 服务器。

在样本主配置文件中还可以看到 listen-on、dump-file、allow-query 和 recursion 等子句。

2）zone 语句

zone 语句定义一个区域块，其中必须说明域名、DNS 服务器的类型和区域文件名等信息，其基本格式为：

```
zone {
        type 子句;
        file 子句;
        其他配置子句;
};
```

（1）type 子句说明了 DNS 服务器的类型，经常使用的参数是 master 和 slave，还可以使用的类型有 hint、stub 和 forward。master 表示此 DNS 服务器为主域名服务器；slave 表示此服务器是辅助服务器。

（2）file 子句指定区域文件的名称，应在文件名两边使用双引号。

比如：

```
zone "hncj.edu.cn." IN {
   type master;
   file "hncj.edu.cn.zone";
};
```

定义了一个名为 hncj.edu.cn 的 zone，它的类型为 master，区文件为 hncj.edu.cn.zone。

3）include 语句

include 语句用于定义主配置文件包含的子配置文件，例如：

```
include "/etc/named.rfc1912.zones";
```

用 include 定义的包含文件可以作为配置文件的一部分，在 DNS 服务器启动时被读取。

从默认的主配置文件 named.conf 和其包含文件 named.rfc1912.zones 可以得知，配置 DNS 服务器还需要使用其他的文件，表 19-1 中列举了与 DNS 服务器配置相关的一些文件。

表 19-1 DNS 配置需要的常用文件

文件类型	文件名	相关说明
主配置文件	/etc/named.conf	用于设置 DNS 服务器的全局参数，并指定区域文件名及其保存路径
包含文件	/etc/rndc.key	密钥文件
缓冲数据库文件	/var/named/named.ca	为缓冲服务器数据库文件，包含了全球 13 个 DNS 根服务器信息，通常不需要手动修改
本地区域文件	/var/named/named.localhost	用于将主机名 localhost 转换为回送 IP 地址
本地回送文件	/var/named/named.loopback	用于将回送 IP 地址转换为主机名 localhost
正向区域文件	在 named.conf 中包含。一般为/var/named/<domain>.zone	用于实现区域内主机名到 IP 地址的正向解析
反向区域文件	在 named.conf 中包含。一般为/var/named/<IP3R>.in-addr.arpa.zone	用于实现区域内 IP 地址到主机名的反向解析

在表 19-1 中，domain 为用户需要建立的域名，比如用户需要解析的域名 hncj.edu.cn，则正向区域文件名为 hncj.edu.cn.zone。所谓“IP3R”是指将需要解析的点分十进制 IP 前 3 节进行逆向排序，比如，若用户需要反向解析的 IP 地址为 211.67.112.x，则反向区域文件为 112.67.211.in-addr.arpa.zone。若觉得这种命名方法使得文件名太长的话，用户也可以在主配置文件中按照易用、易理解的原则将其命名为其他格式，比如 211.67.112.zone。

2．正向区域文件

正向区域文件定义了一个区域的域名、IP 地址等信息，主要由若干个记录组成，下面是典型的正向区域文件内容：

```
$TTL 1D
@     SOA  dns.hncj.edu.cn.  root.hncj.edu.cn. (
                                    2017010101
                                    3H
                                    1H
                                    1W
                                    1H )
      NS   dns.hncj.edu.cn.
      MX   5    gw
      MX   10   mail
dns   IN   A    211.67.112.1
gw    IN   A    211.67.112.20
mail  IN   A    211.67.112.21
www   IN   A    211.67.112.81
news  IN   CNAME    www
```

在该文件中出现的资源记录如下。

1）起始授权记录 SOA

SOA 记录总是处于区域文件中所有记录的最前面，用于表示一个区的开始。每个区域文件中都必须包含一个 SOA 记录，以表示域名服务器所管理的范围，其基本格式为：

```
域名 IN   SOA      主机名    管理员邮件地址(
               序列号; serial
               刷新间隔时间; refresh
               重试间隔时间; retry
               过期间隔时间; expire
               最小存活时间; ttl
               )
```

（1）域名：为用户定义区域的域名，一般情况下为@，表示其值为主配置文件中相应区域的名称。

（2）IN：代表 Internet 类。其他类还有 HS（hersod）、CH（ChaosNet）等，目前使用最多的是 IN 类，也是默认类，可以省略。

（3）SOA：表示起始授权类型，其功能是将某区域授权给某台服务器管理。

（4）主机名：是授权 DNS 服务器，它采用的是 FQDN，末尾必须要用“.”结尾。

（5）管理员邮件地址：因为“@”符号在区域文件中有特殊的含义，管理员的电子邮件地址不能使用“@”符号，而使用“.”代替。

（6）serial（序列号）：也称为版本号，用来表示该区域数据库文件的版本大小，最多有 10 位数字。一般情况下，采用类似年月日的表示形式，如 20120101xx，这种写法可方便管理记住最后修改该文件的日期和次数。需要注意的是，新版本号应该比旧版本号要大。当辅 DNS 服务器与主 DNS 服务器同步将比较此字段，以判断是否被更新过。

（7）refresh（刷新间隔时间）：其功能是设定辅 DNS 服务器与主 DNS 服务器同步的间隔或刷新周期，默认单位为秒，可以指定单位为分钟（M）、小时（H）、天（D）、周（W）、月（M）等。

（8）retry（重试时间间隔）：单位同刷新时间间隔，它的功能是指定主、辅 DNS 间的同步频率失败后应该重试同步时间延迟或间隔。如果在 refresh 指定的时间内，辅 DNS 服务器没有与主 DNS 服务器同步成功，那么需要在此间隔内重新尝试一次。一般情况下，重试间隔时间应小于刷新间隔时间。

（9）expire（过期时间间隔）：单位同刷新间隔时间，其功能是设置辅 DNS 服务器缓存信息的有效期。如果在此时间间隔内辅 DNS 始终没有与主 DNS 同步成功，那么辅 DNS 服务器则会认为自己的数据已经过期，进而停止响应客户对该区域的查询。

（10）ttl（存活时间）：单位同刷新间隔时间，其功能是设置每条记录的生存期。该生存期是指 DNS 服务器将本区域中相应的记录应答给客户后，该记录在客户机的缓存中会保留多长时间。

2）名字服务器类型记录 NS

NS 记录用于指明区域中 DNS 服务器的主机名，是区域文件中不可缺少的资源记录。由于其作用在与 SOA 记录相同的域，因此可以不写域名。需要注意的是，指定的主机名采用的是 FQDN，必须要用“.”结尾。比如：

```
@    IN    NS    dns.hncj.edu.cn.
```

3）主机类型的记录 A

A 记录用于指定域名与 IP 地址的相互关系，仅可用于正向区域文件，通常只需写出主机名，而不用写出完整的域名，例如：

```
www     IN    A     211.67.112.81
```

4）别名类型的记录 CNAME

CNAME 记录用于为区域内的主机建立别名，仅用于正向区域文件，常用于一个 IP 地址对应多个不同类型服务器的情况，可以使用 A 记录代替。例如：

```
news    IN       CNAME    www
```

5）邮件交换记录 MX

MX 记录用于指定区域内邮件服务器的域名与 IP 地址的相互关系，仅用于正向区域文件。当区域中有多个邮件服务器时，可以通过 MX 记录来指定邮件服务器的优先级别，数字越小，优先级越高。例如：

```
@          IN        MX        5 gw.hncj.edu.cn.
```

```
@        IN       MX       10 mail.hncj.edu.cn.
```

这说明区域中有两个邮件服务器 gw 和 mail，当用户向域内发邮件时，发送邮件的服务器会优先将邮件投递给 gw.hncj.edu.cn，因为其优先级别为 5，大于 mail 的 10。

3．反向区域文件

在理解了正向区域文件的定义后，可以很容易理解反向区域文件。以下是典型的反向区域文件内容：

```
$TTL 1D
@    SOA  dns.hncj.edu.cn.  root.dns.hncj.edu.cn. (
                    2017010100
                    3H
                    1H
                    1W
                    1H )
     NS   dns.hncj.edu.cn.
1    PTR  dns.hncj.edu.cn.
20   PTR  gw.hncj.edu.cn.
21   PTR  mail.hncj.edu.cn.
81   PTR  www.hncj.edu.cn.
```

将正向区域文件和反向区域文件进行对比可以发现，它们的前两条记录 SOA 和 NS 是相同的，而一般情况下正、反向区域文件中这两条记录是相同的。唯一可以区别的是正向区域文件包含有 A 记录、MX 记录、CNAME 记录，而反向区域文件只包括 PTR 记录。

PTR 类型又称为反向类型或指针类型，是用来定义由 IP 地址到域名翻译记录的。通常只用写出完整 IP 地址的最后一部分，例如：

```
1    IN    PTR  dns.hncj.edu.cn.
```

最后 1 列为域内的一个主机，第 1 列为该主机所对应的 IP 地址的最后 1 段。此行中表明 dns.hncj.edu.cn IP 的主机地址为 1。也就是说，若域 hncj.edu.cn 的网络地址为 211.67.112.X 的话，dns.hncj.edu.cn 的 IP 地址为 211.67.112.1。

注意：BIND 中这样规定，在正、反向区域文件中，每条记录如果最左边或最右边两列的值不是以“.”结尾，那么系统将会自动在该值后面补上@的值，即补上区域名称来构成 FQDN，该区域的名称也是在 DNS 服务器主配置文件 named.conf 中的 zone 里定义的。

说明：按照主配置 named.conf 的注释，用户不要修改主配置文件（其实，主配置文件是可以修改的），当需要添加区块文件定义时，可参照其包含文件格式将需要添加的内容放在包含文件 named.rfc1912.zones 中，Ubuntu 建议添加到 named.conf.local 中。Ubuntu bind9 的主配置文件内容如下：

```
// This is the primary configuration file for the BIND DNS server named.
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
```

```
// If you are just adding zones, please do that in /etc/bind/named.conf.local
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
```

19.3 DNS 服务器配置示例

1．主域名服务器配置示例

任务：建立一个符合以下条件的主域名服务器。

（1）域名注册为 hncj.edu.cn，网段地址为 211.67.112.*。

（2）主域名服务器主机名为 dns.hncj.edu.cn。

（3）要解析的服务器有 dns.hncj.edu.cn、www.hncj.edu.cn，对应的 IP 地址分别为 211.67.112.1 和 211.67.112.3，mail.hncj.edu.cn 为 dns.hncj.edu.cn 的别名记录。

（4）主邮件服务器为 mail.hncj.edu.cn。

配置过程大致如下：

首先根据所用系统和 DNS 服务器的运行方式，确定配置文件的位置，然后进入主配置文件所在配置目录。Ubnutu 下为/etc/bind，红帽系统 chroot 运行方式为/var/named/chroot/etc/，非 chroot 方式为/etc。也可以是主配置中 directory 指定的目录/var/named（红帽）或/var/cache/bind（Ubuntu）。当使用 directory 指定的目录时，主配置文件中的 include 命令可以使用相对路径。

步骤 1：在 named.conf 文件或其包含文件（比如红帽系统的 named.rfc1912.zones，或 Ubuntu 的/etc/bind/named.conf.local）中加入以下内容：

```
zone "112.67.211.IN-ADDR.ARPA." IN {
        type master;
        file "211.67.112.rev";
};
zone "hncj.edu.cn." IN {
        type master;
        file "hncj.edu.cn.zone";
};
```

步骤 2：创建正向域文件 hncj.edu.cn.zone，并加入如下内容：

```
$TTL 1D
@       SOA dns.hncj.edu.cn.  root.hncj.edu.cn. (
                              2017010103
                              3H
                              1H
                              1W
                              1H )
        NS      dns.hncj.edu.cn.
        MX  5       gw
```

```
        MX   10     mail
dns     IN   A      211.67.112.1
gw      IN   A      211.67.112.20
mail    IN   A      211.67.112.21
www     IN   A      211.67.112.81
news    IN   CNAME  www
```

步骤 3：创建逆向域文件 211.67.112.rev，并加入以下内容：

```
$TTL 1D
@       SOA  dns.hncj.edu.cn.  root.dns.hncj.edu.cn. (
                               2017010100
                               3H
                               1H
                               1W
                               1H )
             NS     dns.hncj.edu.cn.
1       PTR  dns.hncj.edu.cn.
20      PTR  gw.hncj.edu.cn.
21      PTR  mail.hncj.edu.cn.
81      PTR  www.hncj.edu.cn.
```

注意，PTR 指针里的名称都要采用 FQDN。

步骤 4：检查并修改新建文件的所有者和权限。

在红帽系统下，与 DNS 相关的文件的主、组均为 named，在 Ubuntu 下的主、组均为 bind，用户可以使用以下命令管理之。

```
#chown named:named 211.67.112.rev hncj.edu.cn.zone          #红帽系统
#chown bind:bind 211.67.112.rev hncj.edu.cn.zone            #Ubuntu
#chmod ug+rw hncj.edu.cn.zone 211.67.112.rev
```

步骤 5：查看配置文件是否有错误。

```
#named-checkconf                                             #检查 named.conf 文件语法
#named-checkzone hncj.edu.cn /var/named/hncj.edu.cn.zone     #检查区域配置文件语法
```

步骤 6：启动/重启服务或重新加载配置。

```
#systemctl restart | reload named-chroot      #红帽系统：以 chroot 方式运行
#systemctl restart | reload named             #红帽系统：以非 chroot 方式运行
#systemctl restart | reload bind9             #Ubuntu 系统
```

步骤 7：查看 named-chroot 状态。

```
#systemctl status named-chroot                #红帽系统：以 chroot 方式运行
#systemctl status named                       #红帽系统：以非 chroot 方式运行
#systemctl status bind9                       #Ubuntu 系统
```

根据显示信息可以了解 DNS 服务器是否成功启动。因为，有时即使屏幕显示已正常启动守护进程，但 DNS 服务器仍然可能存在问题，需要查看运行状态或者日志文件

/var/log/messages 来判断是否有错误发生。比如，可能会出现守护进程运行正常，由于区域文件权限问题无法正常解析的情况。

2．配置辅域名服务器

任务：建立一个注册名为 hncj.edu.cn，IP 地址为 211.67.112.1 的服务器。

步骤 1：在辅域名服务器的主配置文件 named.conf 或其包含文件 named.rfc1912.zones 中加入以下内容。

```
zone "hncj.cn." IN {
        type slave;
        file "hncj.edu.cn.zone";
        masters {    211.67.112.1 port 53;
    };
};
```

步骤 2：参照上例步骤 4，检查新建文件的所有者和权限，若不正确应根据具体情况配置。

步骤 3：启动/重启服务，并检查服务的运行状态信息。

步骤 4：查看工作目录中是否已经自动复制了区域文件。

启动区域传输机制的情况只有 3 种，一是辅域名服务器刚启动，二是 SOA 记录中的刷新间隔到达，三是 master DNS 设置了主动通知辅域名服务器数据有变化。以上 3 种情况之一发生后，辅域名服务器会主动与主域名服务器进行数据同步。

19.4 测试 DNS 服务器

不论 DNS 是哪种类型，也无论采用哪种方法运行 DNS 服务，都应该测试该服务是否能够正常工作。目前最常用的 DNS 服务测试工具有 nslookup、host 和 dig 三种。我们已经在前面介绍了 nslookup 和 host，这里再介绍一下 dig。

dig（domain information groper）是 Linux 中的域名解析工具，功能比 nslookup 和 host 要强一些，dig 的常用方法为：

dig [-h] [@server] [-x IP_addr] [type| -t type] [host] [dopt]

-h 用于显示帮助信息；@server 用于指定 DNS 服务器；host 用于指定要查询的主机名；-x IP_addr 用于逆向查询 IP_addr 所对应的主机名；type 用于指定查询类型，可取的值有 ANY，A，MX，NS，SOA，HINFO，AXFR(zone 数据传输)，IXFR=N（zone 数据的增量传输）等，默认为 A；dopt 用于控制 dig 的输出，可用值有很多，如+noshort/+short（长/短格式），+noall/+all（无/所有），+nocomments/+comments（无/有注释），+nostats/+stats（无/有统计信息），+noquestion/+question（无/有 QUESTION 节），+noanswer/+answer（无/有 ANSWER 节），+noauthority/+authority（无/有 AUTHORITY 节），+noadditional/+additional（无/有 ADDITIONAL 节）。

dig 的应用示例如下：

1．正向查询

```
#dig dns.hncj.edu.cn
; <<>> DiG 9.10.4-P8-RedHat-9.10.4-4.P8.fc25 <<>> dns.hncj.edu.cn
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10732    //头部信息
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; OPT PSEUDOSECTION:                  //虚选项部
; EDNS: version: 0, flags:; MBZ: 0x0005, udp: 4096
;; QUESTION SECTION:                   //查询部分
;dns.hncj.edu.cn.         IN     A

;; ANSWER SECTION:                     //回复部分
dns.hncj.edu.cn.  5       IN     A       211.67.112.1

;; AUTHORITY SECTION:                          //权威机构部分
hncj.edu.cn.              5       IN      NS     dns2.hncj.edu.cn.
hncj.edu.cn.              5       IN      NS     dns.hncj.edu.cn.

;; ADDITIONAL SECTION:                         //附加部分
dns2.hncj.edu.cn.5        IN     A       61.163.226.16

;; Query time: 0 msec
;; SERVER: 192.168.75.2#53(192.168.75.2)
;; WHEN: Tue May 09 16:00:40 CST 2017
;; MSG SIZE    rcvd: 109
```

2．反向查询

```
#dig -x 211.67.112.1                           #反向查询需要加上-x 参数
; <<>> DiG 9.10.4-P8-RedHat-9.10.4-4.P8.fc25 <<>> -x 211.67.112.1
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33486
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;1.112.67.211.in-addr.arpa.   IN      PTR

;; ANSWER SECTION:
```

```
1.112.67.211.IN-ADDR.ARPA. 86400 IN     PTR  dns.hncj.edu.cn.

;; AUTHORITY SECTION:
112.67.211.IN-ADDR.ARPA. 86400     IN     NS     dns.hncj.edu.cn.

;; ADDITIONAL SECTION:
dns.hncj.edu.cn.  86400       IN     A      211.67.112.1

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue May 09 16:02:16 CST 2017
;; MSG SIZE    rcvd: 138
```

3．其他查询

（1）查找 hncj.edu.cn 的 A 记录。

```
#dig www.hncj.edu.cn A +noall +answer
#dig www.hncj.edu.cn A +noall +answer +short
```

（2）查找 hncj.edu.cn MX 记录的列表。

```
#dig hncj.edu.cn MX +noall +answer
```

（3）查找 hncj.edu.cn 的权威 DNS。

```
#dig hncj.edu.cn NS +noall +answer
```

（4）查询上面所有的记录。

```
#dig hncj.edu.cn ANY +noall +answer
```

19.5　DNS 与防火墙及 SELinux 的关系

19.5.1　DNS 与防火墙的关系

DNS 服务器使用的服务为 domain，端口号为 53，在设置防火墙时，应该允许该端口通过，设置方法如下。

1．firewalld

```
#firewall-cmd --permanent --list-all                 #显示已开放端口
#firewall-cmd --add-port=53/udp –permanent           #添加端口
#firewall-cmd --reload                               #重新加载配置
```

2．ufw

```
#ufw status                                          #检查规则状态信息
#ufw app list                                        #检查应用程序信息
```

```
#ufw allow domain                                  #允许 DNS 服务
```

19.5.2　DNS 与 SELinux 的关系

1．布尔变量

SELinux 对 DNS 所使用的控制变量有 named_tcp_bind_http_port（用于控制是否允许 BIND 绑定 Apache 端口）和 named_write_master_zones（用于控制对主域文件的写操作）两个。

2．类型

与 DNS 相关的文件和目录标签在 SELinux 中有 7 种类型，参见表 10-9。查看文件和目录的标签，可使用命令：

```
#ls -Z
```

3．DNS 与 SELinux 的关系

目录/var/named/slaves、/var/named/dynamic 和/var/named/data 的默认权限是允许区域文件更新和传输。目录/var/named/被贴上 named_zone_t 标签。如果区域文件被贴上 named_zone_t 标签，则应该将 named_write_master_zones 设置为 on，以发挥 SELinux 的保护作用，且允许区域文件的自动传输和自动更新。还要保证/var/named/目录对 named 用户和 named 组具有读、写和执行权。

如果/var/named/目录被贴上了错误标签，这时可以通过命令

```
#restorecon -R /var/named
```

将它的权限标签恢复到默认值（参见 10.5 节）。

习题 19

1．试述 DNS 服务器的启动管理方法，写出相关命令。
2．DNS 服务器主要有哪几种类型？
3．简述 DNS 查询模式及解析过程。

实验 19

配置一个主 DNS 服务器，要求满足如下条件：
（1）域名为 Linux. UNIX.hncj，IP 地址为 192.168.50.1；
（2）在域内设有主机 www、mail、ftp 和 bbs，IP 地址分别为 1、2、3 和 4；
（3）要求能够正向和逆向解析；
（4）建好之后，要使用相关工具软件进行测试。

参 考 文 献

[1] [美]Christopher Negus著．王净，田洪译. Linux 宝典（第 9 版）．北京:清华大学出版社，2016.

[2] [美]Mark G.Sobell 著．李洋，等译. Fedora 和 Red Hat Enterprise Linux 实用指南（第 6 版）（上、下册）. 北京:电子工业出版社，2013.

[3] [英]Neil Matthew,Richard Stones 著．陈健，等译. Linux 程序设计（第 4 版）．北京:人民邮电出版社，2010.

[4] [英]Neil Matthew，等著．叶小虎，等译. Linux 高级编程．北京:机械工业出版社，2002.

[5] [美] Eric S. Raymond 著．姜宏，等译. UNIX 编程艺术．北京:电子工业出版社，2012.

[6] [美] 罗宾，比博著. O'Reilly Taiwan 公司译. Shell 脚本学习指南．北京:机械工业出版社，2009.

[7] [美]布鲁姆等著．门佳，等译. Linux 命令行与 shell 脚本编程大全（第 3 版）．北京:人民邮电出版社，2016.

[8] Alfred V. Aho, Brian W. Kernighan, Peter J. Weinberger, The AWK Programming Language, Addison-Wesley Pub. Co, 1988.

[9] Arnold Robbins. Effective awk Programming. O'Reilly Media, 2001.

[10] [美]Steve Suehring 著．王文烨译. Linux 防火墙（第 4 版）．北京:人民邮电出版社，2016.

[11] 邵国金. UNIX 系统与软件开发．北京:中国铁道出版社，2013.

[12] http ://www.fedoraproject.org/, https://getfedora.org/.

[13] http://docs.fedoraproject.org/.

[14] https://download.fedoraproject.org/pub/fedora/linux/releases/24/.

[15] http://www.centos.org/.

[16] http://www.centoscn.com/.

[17] http://www.redhat.com/，https://www.redhat.com/zh/.

[18] https://access.redhat.com/documentation/en/red-hat-enterprise-linux/.

[19] https://access.redhat.com/documentation/zh_cn/red-hat-enterprise-linux/.

[20] http://www.ubuntu.com/https://help.ubuntu.com/.

[21] http://www.debian.org/.

[22] http://www.SuSE.com/.

[23] http://www.redflag-linux.com/.

[24] http://www.cs2c.com.cn/.

[25] http://www.gnu.org/ .

[26] http://www.gnu.org/software/.

[27] http://www.gnu.org/software/gawk/ .

[28] http://www.gnu.org/software/gawk/manual/.

[29] http://www.gnu.org/software/bash/.

[30] http://www.gnu.org/software/bash/manual/ .

[31] http://vdisk.weibo.com/s/mtEZb/.

[32] http://www.gnu.org/software/sed/.

[33] http://www.gnu.org/software/sed/manual/.
[34] http://hadoop.apache.org/.
[35] http://hadoop.apache.org/docs/current/.
[36] https://docs.openstack.org/.
[37] https://docs.openstack.org/zh_CN/.